BERT-Based Multi-Task Learning for Natural Language Understanding

Ray Ortigas
Stanford Center for Professional Development
rortigas@stanford.edu
(CA Mentor: Jingwen Wu, Stanford University)

Abstract

Training models for natural language understanding is hard. Training a unified model to handle multiple tasks well: even harder. When developing our own BERT-based deep neural network using multi-task learning, we encountered challenges training models that performed better than single-task deep learning models we trained on the same tasks. But ultimately, with the help of (1) augmented sentence models with BERT as foundation, (2) contrastive learning to generate better-distributed embeddings in latent space, and (3) gradient surgery to harmonize conflicting objectives, we trained a multi-task learner which achieved a top-10 result on the CS224N Default Final Project test leaderboard, out of over 100 entries. In this paper, we review the techniques we learned in pursuit of our goal: building a multi-task learner that could generate effective, general-purpose embeddings for natural language understanding.

1 Introduction

For our project, we trained a single deep neural network to handle three natural language understanding (NLU) tasks:

- **sentiment classification**, as represented by the Stanford Sentiment Treebank, SST-5 (Socher et al., 2013),
- **paraphrase detection**, as represented by Quora Question Pairs [1], QQP, and
- **semantic textual similarity evaluation**, as represented by the SemEval benchmark, STS-B (Cer et al., 2017).

When approaching these challenges, we used BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) for our foundational embeddings, tuning these representations through multi-task learning. Invoking (Caruana, 1997), our goal was to generate embeddings that could effectively support all of our tasks:

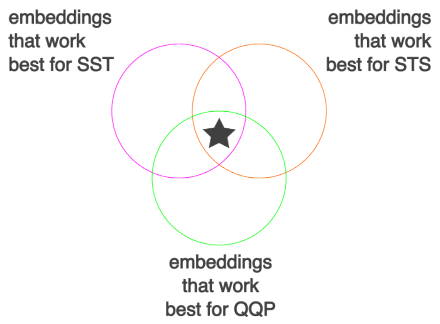

embeddings that work best for SST
embeddings that work best for STS
embeddings that work best for QQP

Figure 1: What we want from multi-task learning for natural language understanding (the star).

[1] formerly at data.quora.com/First-Quora-Dataset-Release-Question-Pairs

While BERT embeddings provide an excellent starting point, we explored several extensions for improving them through our multi-task learner:

- SBERT (Sentence-BERT) sentence embeddings using conjoined BERT networks, as conceived by Reimers and Gurevych (2019), as well as other neural network architectures such as convolutional neural networks (CNNs) and bidirectional long short-term memory (Bi-LSTM) models.
- SimCSE (Simple Contrastive Learning of Sentence Embeddings), as proposed by Gao et al. (2021), for better representing contrast in sentences, by generating embeddings more spread out in latent space.
- PCGrad (Projecting Conflict Gradients), a gradient surgery technique proposed by Yu et al. (2020) to align gradients constructively for multi-task learning.

2 Related Work

2.1 BERT

When Devlin et al. (2019) created BERT, they were part of a wave of improvements in deep learning for NLU:

- **Transformers.** Vaswani et al. (2017) conceived the Transformer architecture to address the limitations of RNNs (recurrent neural networks) such as a LSTM (long short-term memory) neural networks. They constructed an attention mechanism which permitted weighted, parallelized processing of longer-range dependencies than an LSTM can handle, avoiding the sequential processing that an RNN requires, while retaining the capability to track the context needed for understanding larger sequences and passages of text.
- **Bi-directionality.** Despite the above limitations with RNNs, Peters et al. (2018) leveraged multi-layer bi-LSTMs (bi-directional LSTMs) for ELMo. They found that such models could learn rich language representations at varying levels of abstraction: lower levels encode syntax, while higher levels capture context. Furthermore, and crucially, all levels learn by traversing text sequences in both directions. The learned representations could then be used in task-specific architectures downstream, e.g. in conjunction with a BCN (biattentive classification network)-based design as described by McCann et al. (2017) for sentiment classification.
- **Generative pre-training.** Understanding the benefits of pre-training for language models (Mikolov et al., 2013; Pennington et al., 2014), Radford et al. (2018) used both approaches to develop GPT, leveraging transformers to attend to past words for context, as it processed sentences left-to-right. However, in contrast to Peters et al. (2018), who pre-trained ELMo on a one-billion word dataset whose sentences were shuffled, Radford et al. (2018) elected to not shuffle sentences in their similarly sized pre-training dataset. Consequently GPT could generatively model the long-range context and structure that was preserved during pre-training, leading to improved discriminative fine-tuning, for downstream text classification or text prediction.

BERT combined transformers with the best aspects of bi-directionality (like ELMo, while avoiding its use of RNNs and task-specific architectures) and more carefully considered pre-training and generalizability (like GPT, while avoiding its processing of sentences only left-to-right) (Devlin et al., 2019). It is on BERT's foundational embeddings, that we built our project.

2.2 Multi-Task Learning

When Radford et al. (2019) released GPT-2, they asserted that "language models are unsupervised multitask learners". Starting with the original work of Caruana (1997) and Ruder (2017)'s review many years later, we consider why that might be the case:

- **Implicit data augmentation and regularization.** Models handling the QQP and STS tasks for example, likely depend on some language constructs and concepts that underlie paraphrase detection and semantic similarity evaluation, which could be valuable for a neural network to learn as hidden features. The datasets for these tasks, however, come from vastly different sources. A model trained to one of these tasks will likely overfit the noise for the task's dataset (Ruder, 2017); in contrast, a model trained to handle both tasks, can effectively average out the noise to learn hidden features (Caruana, 1997) in a more generalized fashion, while leveraging the tasks' combined datasets for both tasks.

- **Feature selection and eavesdropping.** From implicitly augmented datasets, a multi-task learner can pick and choose which inputs and hidden features matter for a given task. Occasionally however, a feature which is useful to two tasks, say SST and QQP for example, may be harder to learn for SST. Because of multi-task learning our model still has the opportunity to learn the hidden features through QQP, and apply what it's learned, when subsequently handling SST.
- **Representation bias.** Done correctly, multi-task learning should bias the model towards learning hidden representations that are mutually beneficial to the tasks at hand. For SST, QQP, and STS, we would like to find this sweet spot, as depicted in the Venn diagram earlier above.

3 Approaches

3.1 Sentence Modeling

For multi-task learning, our unified model handled all three tasks, whose respective heads/networks had a shared BERT base. Furthermore, we also considered additional extensions to improve these foundations:

- **Sentence-Based Representations.** When using BERT for tasks involving pairs of sentences (like paraphrase detection and semantic textual similarity evaluation), Devlin et al. (2019) would use BERT to process the pair as one input, using token type embeddings to distinguish the sentences. While this approach worked in practice for downstream classification tasks, Reimers and Gurevych (2019) explored the utility of individual sentence embeddings with SBERT, feeding each sentence in the pair individually to BERT, and processing the sentences with twin networks, conjoined at the classification head (which we'll refer to as a conjoined network for the rest of this paper). The subsequent modularity could then help with use cases like clustering and retrieval, as individual sentence embeddings could then be compared straightforwardly with similarity measures such as cosine similarity. To represent an individual sentence, SBERT takes the average of the embeddings returned by the last BERT layer for all the sentence's tokens.
- **Bidirectional LSTMs.** Although SBERT is intuitive for processing sentences individually, it loses the benefits of self-attention that BERT can apply to two sentences when they are fed into BERT as a pair, which effectively achieves cross-attention for the two sentences. To compensate for the lost context, we used BiLSTMs in conjunction with SBERT embeddings. With more time, we would've explored BCNs (by McCann et al. (2017), as noted earlier) and other techniques employing attention.
- **Convolutional Neural Networks.** CNNs have been used for sentence classification, as described by (Chen, 2015). This technique eventually was superseded by the Transformer architecture and descendents like BERT. Still, for our project, we experimented with CNNs to group tokens into unigrams, bigrams, trigrams, and fourgrams (units which are still considered by sentence quality benchmarks like BLEU (Papineni et al., 2002)). Using dropout (Srivastava et al., 2014) and max-pooling on these n-grams, we attempted to extract interesting features that could be useful for sentiment detection, for example. We also fed these n-grams to our BiLSTMs in the hopes of benefitting from processing of these progressively coarser-grained inputs.

For the rest of this paper, we will refer to our augmented sentence model as SBERT + CNN-BiLSTM. For each sentence, it generates an embedding which is a concatenation of the average pooled embeddings from SBERT, the max pooling results from our CNN for the SBERT embedding, and the hidden tokens from our BiLSTM for the SBERT embedding.

3.2 Task Modeling

For our task challenges, we proceeded to use SBERT + CNN-BiLSTM in conjunction with original BERT-based approaches:

- **Sentiment Classification.** When predicting a sentence's sentiment amongst two or more classes, we used a classification head backed by a simple linear layer. After applying dropout to our

SBERT + CNN-BilSTM embeddings, we projected the result to obtain a logit for each clas. We then replaced Our training objective was to minimize on cross-entropy loss. The baseline BERT approach is to use the BERT `[CLS]` token embedding, which is also effectively a pooled output.

- **Paraphrase Detection.** When predicting whether two sentences are paraphrases of each other, we used a classification head backed by a small network. Let $t^{(1)}$ and $t^{(2)}$ be the sentences' respective SBERT + CNN-BiLSTM embeddings in $\mathbb{R}^h$. Using W_q to denote a projection matrix in $\mathbb{R}^{3h \times 1}$ and **;** to denote concatenation, our training objective was to minimize the sigmoid cross-entropy loss on the logit $W_q([t^{(1)}; t^{(2)}; |t^{(1)} - t^{(2)}|])$ (Reimers and Gurevych, 2019). We found this approach mapped intuitively to the starter code, but as we noted, it differed from Devlin et al. (2019)'s processing of a single `[CLS]` embedding for the two sentences concatenated. We found that the latter approach was effective for paraphrase detection, so we ensembled it with our SBERT + CNN-BiLSTM model by combining scores from the two models. Our training objective was to minimize on cross-entropy loss.
- **Semantic Textual Similarity.** Here, we also ensembled SBERT + CNN-BiLSTM with BERT. For the two sentences' respective SBERT + CNN-BiLSTM embeddings, we took their cosine similarity. Meanwhile, with the single `[CLS]` embedding for the two sentences concatenated returned by BERT, we projected it to obtain a score, then adding the cosine similarity from SBERT + CNN-BiLSTM to obtain a logit. Our training objective was to minimize the mean-squared error loss on the logit.

3.3 Additional Techniques

- **Cross-domain pre-training.** We explored using the Stanford Natural Language Inference (SNLI) Bowman et al. (2015) and Multi-Genre Natural Language Inference (MNLI) Williams et al. (2018) corpora. Looking past specific benchmarks and challenges such as SST-5, QQP, and STS, they wanted datasets that could help language models evolve towards better generalizability. Observing that NLU depended on a fundamental ability to infer entailment and contradiction from language they sourced SNLI and MNLI more intentionally from annotators (e.g. by providing carefully crafted instructions and scenarios for annotation), at scale, so that models could learn from a wealth of quality examples. We used the concatenation of MNLI and SNLI as provided by Hugging Face[2].
- **Contrastive learning.** We also pre-trained our models with contrastive learning of embeddings (Gao et al., 2021). If our model could better represent contrast in sentences–by generating embeddings more spread out in latent space–it might also better identify similarity in sentences. For a given mini-batch of N examples, for the ith example, a contrastive loss is calculated for input h_i as $\ell_i = -\log \frac{e^{\text{sim}(h_i, h_i^) + / \tau}}{\sum_{j=1}^{N} e^{\text{sim}(h_i, h_j)^+ / \tau}}$, where h_i^+ is semantically related to h_i (e.g. a known entailment or paraphrase for a given sentence), and τ is a temperature parameter. For our experiments, to obtain h_i^+ we used positive examples from QQP, as well as pairs from STS whose similarity score was greater than 4 (on a scale of 1 to 5).
- **Gradient surgery.** With PCGrad (Yu et al., 2020), using projection of conflicting gradients (i.e. those with negative dot products), we could align gradients constructively to improve learning for all tasks, helping us better reason about convergence and when to stop training, so that our models can perform and generalize well. With two reference implementations from Yu et al. (2020) and Tseng (2020), we wrote a from-scratch implementation of our own. A code listing is provided in the appendix.

4 Experiments

4.1 Task Datasets and Evaluation

Briefly recapping the default project's datasets and related evaluation metrics for our models:

Stanford Sentiment Treebank (SST). For sentiment classification over SST, we evaluated on overall accuracy across all examples, each from one of five classes ranging from negative to positive.

Quora Question Pairs (QQP). For paraphrase detection over QQP, we evaluated on binary classification accuracy.

[2]https://huggingface.co/datasets/sentence-transformers/all-nli

SemEval STS Benchmark. For sentence similarity over STS, we evaluated on the Pearson correlation of predicted similarity values with true similarity values, which are continuously-valued from 0 to 5.

4.2 Setup and Implementation Details

As instructed, we initialized the BERT bases of our models with provided pretrained embeddings. We then trained our multi-task learners for at most three epochs. For each training run, we chose the resulting model's parameters based on the epoch where it achieved the highest overall score (using a formula based on the respective evaluation metrics for our tasks), w.r.t. the dev splits of our data.

- PyTorch with one Nvidia T4 GPU. We used g4dn.xlarge GPU instances from Amazon Web Services.
- Adam optimizer. We used the Adam optimizer we implemented, following the provided pseudocode as instructed.
- Learning rate 1e-5. We played around with this setting as well as layer-wise learning rate decay,
- Weight decay 0.01. Out of caution we set this value to regularize our model and avoid overfitting, and just defaulted to what Devlin et al. (2019) used for BERT.
- Dropout 0.1. We also used this setting to regularize our model, also defaulting to what Devlin et al. (2019) used for BERT.
- Batch size 16. We chose this value mainly to speed up training time while not running out of GPU memory. When using PCGrad, we used a partial batch size of 4 to give different tasks opportunities from uniform random sampling (with replacement) to participate in a training step
- Mixed precision training. We used this also to speed up training time, as recommended in lecture, via Pytorch `autocast`.[3]

Our models took about five to six hours overall for each training run of three epochs, taking double the time when using PCGrad.

4.3 Leaderboard Results

Table 1: Multi-task learning results on test data.

test results	overall	SST	QQP	STS
our result (tied-9th)	.792	.529	.905	.881
leaderboard best	.806	.556	.912	.902

Out of over 100 entries, we placed in the top 10, with an overall score of .792 as of this writing (on the last day of submissions). The results were a pleasant surprise, as going into the test session, we had placed in the top 20 with our best dev leaderboard entry, with a lower overall score of .789. It seems that we built a model that did not overfit the training or dev data, and ultimately generalized well, at least to the unseen test set.

After testing out various combinations of techniques, our entry employed:

1. Augmented sentence models, starting with SBERT and additionally applying CNN/BiLSTM.
2. Contrastive sentence embeddings, using supervised SimCSE pre-training with QQP and STS.
3. Projection of conflicting gradients, via PCGrad.

5 Analysis

5.1 Single-Task Learning

When building our multi-task learner, we also looked at the performance of our approaches when used with single-task learners:

[3]`https://pytorch.org/docs/stable/amp.html#torch.autocast`

- **SST.** Over 10 trials for SBERT + CNN-BiLSTM as applied to single-task SST, we achieved a mean accuracy of .523 ± .008 standard deviation. This result represented an improvement over averaging the tokens from the hidden layer, which achieved .520 ± .010. It also improved upon the multiclass accuracy baseline of .515 ± .004 provided with the project starter code, assuming the approach of taking the `[CLS]` token from BERT, and projecting it after dropout to obtain a logit. Applying weight decay of 0.01 and dropout probability of 0.1 to the baseline approach, we were able to achieve .520 ± .007.
- **QQP.** As QQP was an expensive experiment to run to discover maximum performance when training with the entire dataset, we looked at data points incidentally collected during model development. On a recent set of single-task runs, our combined ensemble of BERT with SBERT + CNN-BiLSTM was able to achieve .905 accuracy. This was comparable to the .900 from BERT alone. Meanwhile, SBERT + CNN-BiLSTM alone could only achieve .875 accuracy.
- **STS.** Over 10 trials for our ensemble of BERT with SBERT + CNN-BiLSTM as applied to single-task STS, we achieved a mean accuracy of .867 ± .003 standard deviation. This appeared to improve upon the .864 ± .004 measured for just BERT. Remarkably, SBERT + CNN-BiLSTM alone was much worse on a single trial, with .777, so low that we take it as representative. However, combining this relatively much weaker learner with BERT, may have given us something akin to boosting.

5.2 Multi-Task Learning

Having reviewed our single-task approaches individually, we now return to discussing our multi-task learner. Table 2 shows our model's results on the dev dataset, both for our model submitted to the test leaderboard as well as for the model after undergoing various ablations:

Table 2: Multi-task learning results on dev dataset. Our best model on the dev dataset is in bold, and was used for submission to the test leaderboard.

sentence representation	training strategy	SST	QQP	STS	overall
SBERT + CNN-BiLSTM	initial	.521	.898	.858	.7827
	NLI	.507	.902	.852	.7783
	SimCSE (QQP/STS)	.513	.903	.876	.7847
	SimCSE (SST/QQP/STS)	.516	.903	.877	.7858
	NLI + SimCSE (QQP/STS)	.495	.902	.851	.7742
	PCGrad	.526	.898	.859	.7845
	PCGrad + NLI	.520	.900	.873	.7855
	PCGrad + SimCSE (QQP/STS)	**.523**	**.905**	**.875**	**.7885**
	PCGrad + SimCSE (SST/QQP/STS)	.515	.903	.881	.7862
	PCGrad + NLI + SimCSE (QQP/STS)	.514	.902	.868	.7833
BERT	initial	.516	.905	.871	.7855
	PCGrad	.511	.903	.878	.7855
	PCGrad + SimCSE (QQP/STS)	.513	.906	.873	.7852
	PCGrad + SimCSE (SST/QQP/STS)	.510	.903	.872	.7830
	train SST then STS then QQP	.397	.901	.788	.7307
	train QQP then STS then SST	.520	.888	.865	.7802

By a few percentage points, our models based on SBERT + CNN-BiLSTM outperformed our models based on BERT. We also tried a second seed and saw similar results, though our time and resources were limited, so we did not get to exhaustively try all the combinations we wanted to. Still, we offer a few potential reasons why our SBERT + CNN-BiLSTM approach worked:

- **Layering CNN-BiLSTM on SBERT, for improvement on SST.** As noted previously when reviewing single-task SST performance, our CNN-BiLSTM implementation outperformed, by a few percentage points, the baseline BERT approach of using the `[CLS]`/pooled output as features for a classifier layer. This outperformance was observed over 10 trials, which had similar variance.

Our best result from multi-task learning managed to extract the mean performance from our single-task trials (.523) and exceeded that on the test set (.529).

- **Supervised contrastive learning on QQP and STS via SimCSE, for improvement on STS.** Combining SBERT + CNN-BiLSTM processing two sentences in a pair, with the baseline BERT approach of processing the `[CLS]`/pooled output for the same two sentences concatenated, we obtained STS results that were almost 3-4 standard deviations above the 10-trial average of the baseline BERT approach. We were also able to reproduce this trend with our second seed. As our ensemble of SBERT + CNN-BiLSTM with the baseline BERT approach was also involved, effectively implemented a kind of boosting that reduced bias/error in the overall multi-task learner.
- **Gradient surgery via PCGrad for overall model improvement.** For our multi-task learner using SBERT + CNN-BiLSTM, PCGrad was able to improve consistently our model's performance. We saw this gain with and without contrastive learning, but we achieved our best result when it was used in combination with SimCSE. We note however that the tradeoff was almost doubled training time, and also that PCGrad did not improve our multi-task learner using the baseline BERT approach.

Supervised pre-training with the combined NLI corpus of MNLI and SNLI did not help our model in our trials. We pre-trained with the whole corpus, and may have unwittingly (over)trained the embeddings in a manner that was counterproductive, at least for the tasks at hand. This result, combined with what we saw for supervised pre-training on QQP and STS using SimCSE, corroborates Sun et al. (2019)'s finding that within-task pre-training is especially beneficial, relative to cross-domain pre-training such as what we attempted with MNLI and SNLI.

The appendix contains confusion matrices for SST and QQP, as well as a scatterplot depicting the correlation for STS, for both the initial BERT model and our best SBERT + CNN-BiLSTM model. We note that the confusion matrix for the initial BERT showing it was relatively stronger at detecting the most positive sentiment, but otherwise accuracy is spread evenly across classes for both models. The appendix also contains example misclassifications our model produced for the challenge tasks. Looking at SST in particular, we see that sometimes models contend with noisy labels ("Hilariously inept and ridiculous" is labeled mostly positive, but the model thinks this is strongly negative, and my judgment is closer to the model's) or challenging parsing ("We have n't seen such hilarity since Say It Is n't So !" is provided with the contractions already separated, and BERT apparently mangles it further as `['[CLS]', 'we', 'have', 'n', "'", 't', 'seen', 'such', 'hi', '##lar', '##ity', 'since', 'say', 'it', 'is', 'n', "'", 't', 'so', '!', '[SEP]']`). With more time, we would love to explore these phenomena, and also find actual hard misclassifications that can inform how to improve our models.

6 Conclusion

With BERT, multi-task learning, and a wealth of techniques from other researchers, we successfully trained a multi-task deep learning model for natural language understanding. While we achieved a top-10 placement on the CS224 Default Project leaderboard, out over 100 entries, we also look forward to improving our understanding of why our models work and when they don't.

7 Ethics Statement

For our project, we focused purely on extracting maximum performance from a multi-task learner for natural language understanding. However, the models we trained could have unintended consequences, from unintended uses, without additional intervention:

1. For example, sentiment classification as applied to a person's communications might have innocuous uses, such as for analyzing and summarizing movie or product reviews. But it could also be misapplied to communications with higher stakes, such as analyzing transcripts of a recorded job interview for the supposed sentiment or enthusiasm of a candidate. On challenging datasets such as what we tackled for this project, even state-of-the-art models can struggle to achieve 60% accuracy with respect to a five-point sentiment scale. When considering such high uncertainty, along with potential for misuse, we might mitigate our models' risks by withholding them from usage altogether.

2. For another example, a model that performs paraphrase detection and semantic textual similarity evaluation could be used to help people who are non-expert in a topic discover ways to improve or articulate their understanding of the topic. This usage could leverage generative or discriminative aspects of the model. While many times such usage is beneficial, if it is applied to fields where precision and accuracy are critical, such as in law or in medicine, then a model, in spite of the high accuracy demonstrated by this project and others', may occasionally lead to misjudgments or misinterpretations that can have serious consequences. A possible technical mitigation to this risk is to develop another model which analyzes inbound queries to a system leveraging the model; if the queries are classified as sensitive or requiring care (as for law or medicine), the system can refuse to address the queries.

8 Acknowledgments

While I am the sole author of this paper, I am thankful for the mentorship of Jingwen Wu, who advised me on this project.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28:41–75.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

Yahui Chen. 2015. Convolutional neural network for sentence classification. Master's thesis, University of Waterloo.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. *Advances in neural information processing systems*, 30.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18*, pages 194–206. Springer.

Lisa Torrey and Jude Shavlik. 2009. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques: Algorithms, Methods, and Techniques*, page 242.

Wei-Cheng Tseng. 2020. Weichengtseng/pytorch-pcgrad.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836.

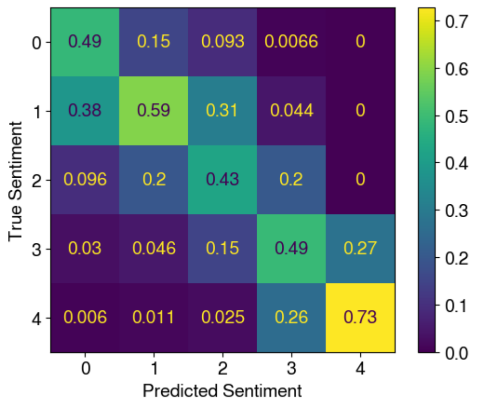
A Appendix

A.1 PCGrad Code Listing

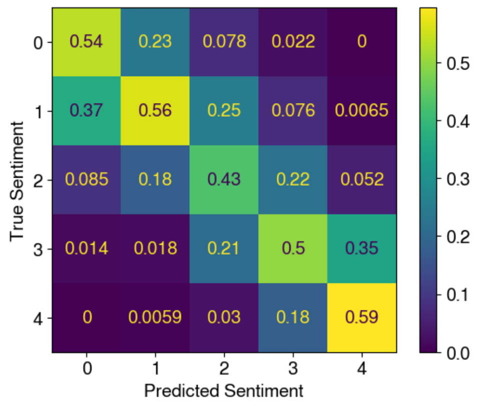
```
# PCGrad implementation
# Ray Ortigas
#
# This module attempts to implement the original algorithm in
#
# https://github.com/tianheyu927/PCGrad
#
# @article{yu2020gradient,
#   title={Gradient surgery for multi-task learning},
#   author={Yu, Tianhe and Kumar, Saurabh and Gupta, Abhishek and Levine, Sergey and Hausman, Karol and Finn, Chelsea},
#   journal={arXiv preprint arXiv:2001.06782},
#   year={2020}
# }
#
# using their TensorFlow implementation as a reference implementation, along with
#
# https://github.com/WeiChengTseng/Pytorch-PCGrad
#
# @misc{Pytorch-PCGrad,
#   author = {Wei-Cheng Tseng},
#   title = {WeiChengTseng/Pytorch-PCGrad},
#   url = {https://github.com/WeiChengTseng/Pytorch-PCGrad.git},
#   year = {2020}
# }
#
# as another reference implementation (whose tests were used to verify basic results).

import random
from typing import List, Optional

import numpy as np
import torch
from torch.optim import Optimizer


def pcgrad_backward_and_step(
    losses: List[torch.Tensor],
    optimizer: Optimizer,
    device: Optional[torch.device] = None,
):
    # Only track gradients of parameters which require gradients.
    params = [
        param
        for group in optimizer.param_groups
        for param in group["params"]
        if param.requires_grad
    ]

    # grads[k][p] is the gradient for the pth parameter, w.r.t. loss/task k.
    # It could be None, if there is no gradient w.r.t. the loss/task.
    grads: List[Optional[torch.Tensor]] = []

    # In the paper, elements of gs are the g_k for all k.
    # gs[k][p] is the flattened gradient for the pth parameter, w.r.t.
    # loss/task k, if there is a gradient; zeros of the same shape otherwise.
    gs: List[torch.Tensor] = []

    # In the paper, elements of g_pcs are the g_k^PC for all k.
    # Each g_k^PC is initialized as g_k and then modified by Algorithm 1.
    g_pcs: List[torch.Tensor] = []

    # For each loss, backpropagate to obtain each loss/task’s gradients,
    # packing theminto a combined gradient for the loss/task.
    for k, loss in enumerate(losses):
        optimizer.zero_grad(set_to_none=True)
        loss.backward(retain_graph=True)

        grads.append(
            [
                p.grad.detach().clone().to(None) if p.grad is not None else None
                for p in params
            ]
        )
        gs.append(
            torch.cat(
                [
                    (
                        grad.flatten().detach().clone()
                        if grad is not None
                        else torch.zeros(np.prod(param.shape))
                    ).to(device)
                    for (param, grad) in zip(params, grads[k])
                ]
            )
        )
        g_pcs.append(gs[k].detach().clone().to(device))

    # With the flattened/packed per-loss/task gradients, run Algorithm 1.
    optimizer.zero_grad(set_to_none=True)
    for i, loss in enumerate(losses):
        js = list(range(len(losses)))
        random.shuffle(js)
        for j in js:
            if j != i:
                gi_pc = g_pcs[i]
                gj = gs[j]
                gi_pc_gj = gi_pc.dot(gj)
                if gi_pc_gj < 0:
                    # Subtract the projection of gi_pc onto gj.
                    # Dot product yields squared L2 norm.
                    gi_pc -= (gi_pc_gj / gj.dot(gj)) * gj
```

```
# After projecting conflicting gradients, unpack them into the respective parameters,
# using their known shapes.
optimizer.zero_grad(set_to_none=True)
index = 0
for p, param in enumerate(params):
    length = np.prod(param.shape)
    g_pcs_p = [
        g_pcs[i][index : (index + length)]
        .view(param.shape)
        .detach()
        .clone()
        .to(device)
        for i in range(len(losses))
        if grads[i][p] is not None
    ]
    g_pc = (
        torch.stack(g_pcs_p, dim=0).mean(dim=0)
        if g_pcs_p
        else torch.zeros(param.shape).to(device)
    )
    param.grad = g_pc
    index += length

# Step!
optimizer.step()
```

A.2 SST Results



(a) BERT



(b) SBERT + CNN-BiLSTM

Figure 2: SST-5 confusion matrices.

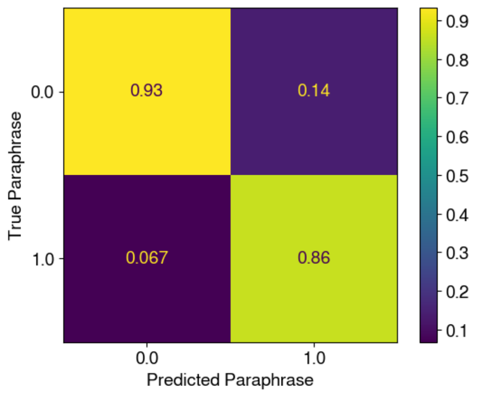
Table 3: Sentence pairs for which model overestimated positive sentiment.

sentence	label	prediction	difference
Confirms the nagging suspicion that Ethan Hawke would be even worse behind the camera than he is in front of it .	0.00	3.00	-3.00
On the whole , the movie lacks wit , feeling and believability to compensate for its incessant coarseness and banality .	0.00	3.00	-3.00
It 's difficult to imagine the process that produced such a script , but here 's guessing that spray cheese and underarm noises played a crucial role .	0.00	3.00	-3.00

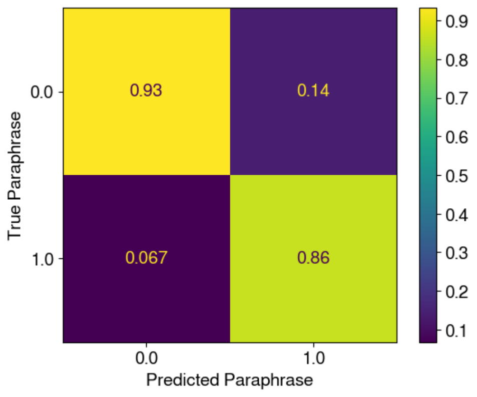
Table 4: Sentence pairs for which model underestimated positive sentiment.

sentence	label	prediction	difference
Hilariously inept and ridiculous .	3.00	0.00	3.00
No screen fantasy-adventure in recent memory has the showmanship of Clones ' last 45 minutes .	4.00	1.00	3.00
We have n't seen such hilarity since Say It Is n't So !	4.00	1.00	3.00

A.3 QQP Results



(a) BERT



(b) SBERT + CNN-BiLSTM

Figure 3: QQP confusion matrices.

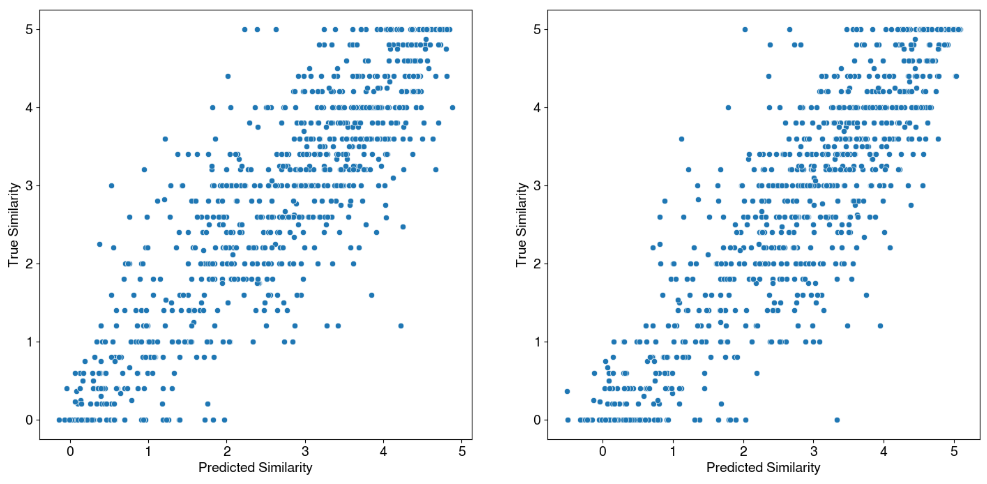
Table 5: Example QQP sentence pairs for which model had false positives.

sentence1	sentence2	label	prediction	difference
How do I lose belly fat and gain muscle mass at the same time? How much treadmill/running is too much to be harmful?	How do I lose belly fat and gain muscle at the same time?	0.00	1.00	-1.00
Which is the best training institute for spring and hibernate in bangalore?	Which is the best training institute for a Spring course in Bangalore?	0.00	1.00	-1.00
How do I stop white hair growing?	What should be done to stop white hair and beard from growing?	0.00	1.00	-1.00

Table 6: Example QQP sentence pairs for which model had false negatives.

sentence1	sentence2	label	prediction	difference
Is insurance a legal scam?	Is insurance a scam?	1.00	0.00	1.00
Where crystal oscillators are used?	What are the uses of a crystal oscillator?	1.00	0.00	1.00
What's the best website to learn extensive programming from?	What is the best website or book to learn a programming language from?	1.00	0.00	1.00

A.4 STS Results



(a) BERT (b) SBERT + CNN-BiLSTM

Figure 4: STS correlation visualizations.

Table 7: Sentence pairs for which model overestimated similarity.

sentence1	sentence2	label	prediction	difference
Work into it slowly.	It seems to work.	0.00	3.34	-3.34
Use of force in defense of person.-A	Use of force by aggressor.	1.20	3.95	-2.75
Libya PM names new cabinet after protests over original	Libyan assembly rejects new cabinet, dismisses PM	1.20	3.49	-2.29

Table 8: Sentence pairs for which model underestimated similarity.

sentence1	sentence2	label	prediction	difference
It’s also a matter of taste.	It’s definitely just a matter of preference.	5.00	2.03	2.97
. What is your definition of "life force"?	What is your definition of "soul"?	3.60	1.12	2.48
A swimmer is doing the back-stroke in the swimming pool.	a person doing the back stroke in a swimming pool	4.80	2.38	2.42