An End-to-End Contrastive Self-Supervised Learning Framework for Language Understanding

Hongchao Fang, Pengtao Xie*
University of California San Diego, USA
p1xie@eng.ucsd.edu

Abstract

Self-supervised learning (SSL) methods such as Word2vec, BERT, and GPT have shown great effectiveness in language understanding. Contrastive learning, as a recent SSL approach, has attracted increasing attention in NLP. Contrastive learning learns data representations by predicting whether two augmented data instances are generated from the same original data example. Previous contrastive learning methods perform data augmentation and contrastive learning separately. As a result, the augmented data may not be optimal for contrastive learning. To address this problem, we propose a four-level optimization framework that performs data augmentation and contrastive learning end-to-end, to enable the augmented data to be tailored to the contrastive learning task. This framework consists of four learning stages, including training machine translation models for sentence augmentation, pretraining a text encoder using contrastive learning, finetuning a text classification model, and updating weights of translation data by minimizing the validation loss of the classification model, which are performed in a unified way. Experiments on datasets in the GLUE benchmark (Wang et al., 2018a) and on datasets used in Gururangan et al. (2020) demonstrate the effectiveness of our method.

1 Introduction

Self-supervised learning (Bengio et al., 2000; Mikolov et al., 2013; Devlin et al., 2019; Radford et al., 2018; Lewis et al., 2020), which learns data representations by solving prediction tasks defined on input data without leveraging human-provided labels, has achieved broad success in NLP. Many NLP-specific self-supervised learning (SSL) methods have been proposed, such as neural language models (Bengio et al., 2000), Word2vec (Mikolov et al., 2013), BERT (Devlin et al., 2019), GPT (Radford et al., 2018), BART (Lewis et al., 2020), and so forth, with various SSL tasks defined. For example, in Word2vec and BERT, the SSL task is predicting the identities of masked tokens based on their contexts. In neural language models including GPT, the SSL task is language modeling: Given a history of tokens, predict the next token.

Recently, contrastive self-supervised learning (He et al., 2020; Chen et al., 2020) has been borrowed from vision domains into NLP and has shown promising success in predicting semantic textual similarity (Gao et al., 2021), machine translation (Pan et al., 2021), relation extraction (Su et al., 2021), and so on. The key idea of contrastive self-supervised learning (CSSL) is: Create augments of original examples, then learn representations by predicting whether two augments are from the same original data example. In existing CSSL approaches, data augmentation and contrastive learning are performed separately. As a result, augmented data may not be optimal for contrastive learning. For example, considering a back-translation (Sennrich et al., 2016)–based augmentation method, if the translation model is trained using news corpora, it is not suitable for augmenting data for movie review data.

In this paper, we aim to address this issue. We propose a four-level optimization framework that performs data augmentation and contrastive learning end-to-end in a unified way, to allow the data augmentation models to be guided by the contrastive learning task and make augmented data suitable for performing contrastive learning. We assume the end task is text classification. Our framework consists of four learning stages. At the first stage, we train four translation models to perform sentence augmentation based on back translation. To account for the fact that translation data used at this stage and text classification data used in later stages have a domain discrepancy, we perform reweighting of translation data; these weights are tentatively fixed at this stage and will

*Corresponding author.

Transactions of the Association for Computational Linguistics, vol. 10, pp. 1324–1340, 2022. https://doi.org/10.1162/tacl_a_00521
Action Editor: Dipanjan Das. Submission batch: 9/2021; Revision batch: 5/2022; Published 11/2022.

be updated at a later stage. At the second stage, we pretrain a text encoder using contrastive learning on augmented sentences created by the translation models. At the third stage, we finetune a text classification model, using the text encoder pretrained at the second stage as regularization. At the fourth stage, we measure the performance of the text classifier on a validation set and update weights of translation data by maximizing the validation performance. Each level of optimization problem in our framework corresponds to a learning stage. These stages are performed end-to-end. Experiments on datasets in the GLUE benchmark (Wang et al., 2018a) and on datasets used in Gururangan et al. (2020) demonstrate the effectiveness of our method.

The major contributions of this paper include:

- We propose a four-level optimization framework to perform contrastive learning (CL) and data augmentation end-to-end. Our framework enables the training of augmentation models to be guided by the CL task and makes augmented data suitable for CL.
- We demonstrate the effectiveness of our method on datasets in the GLUE benchmark (Wang et al., 2018a) and on datasets used in Gururangan et al. (2020).

2 Related Works

2.1 Contrastive Learning in NLP

Recently, contrastive learning has received increasing attention in NLP. Gao et al. (2021) proposed a simple contrastive learning–based sentence embedding method. In this method, the same input sentence is fed into a pretrained RoBERTa (Liu et al., 2019) model twice by applying different dropout masks and the resulting two embeddings are labeled as being similar. Embeddings of different sentences are labeled as dissimilar. Pan et al. (2021) proposed a contrastive learning method for many-to-many multilingual neural machine translation, where contrastive learning is leveraged to close the gap among representations of different languages. Su et al. (2021) developed a contrastive learning method for biomedical relation extraction, where linguistic knowledge is leveraged for data augmentation. Wang et al. (2021) proposed to construct semantically negative examples to perform contrastive learning, for the sake of improving the robustness against semantical adversarial attacks. Pan et al. (2022) proposed to perform contrastive learning on adversarial examples generated by perturbing word embeddings, in order to learn noise-invariant representations.

2.2 Contrastive Self-Supervised Learning in Non-NLP Domains

Contrastive self-supervised learning has been broadly studied recently in other domains besides NLP. Henaff (2020) proposed a contrastive predictive coding method for data-efficient classification. In this method, autoregressive models are leveraged to predict the future in a latent space. Khosla et al. (2020) proposed a supervised contrastive learning method. Data examples having the same class label are made close to each other in the latent space while examples with different class labels are separated farther apart. Laskin et al. (2020) proposed a method to learn contrastive unsupervised representations for reinforcement learning. In Klein and Nabi (2020), a contrastive self-supervised learning approach is proposed for commonsense reasoning.

2.3 Bi-level Optimization

Our framework is a multi-level optimization framework, which is an extension of bi-level optimization (BLO). BLO (Dempe, 2002) has been applied for many applications in NLP, such as neural architecture search (Liu et al., 2018), hyperparameter tuning (Feurer et al., 2015), data reweighting (Shu et al., 2019; Ren et al., 2020; Wang et al., 2020), label denoising (Zheng et al., 2021), learning rate adjustment (Baydin et al., 2018), meta learning (Finn et al., 2017), data generation (Such et al., 2020), and so forth. In these BLO-based methods, meta parameters (neural architectures, hyperparameters, importance weights of training data examples, etc.) are learned by minimizing a validation loss and weight parameters are optimized by minimizing a training loss.

3 Method

In this section, we present our proposed end-to-end contrastive learning framework.

3.1 Overview

We use back-translation (Sennrich et al., 2016) to perform data augmentation of sentences. Then

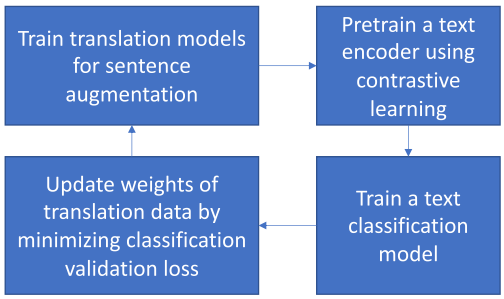

Figure 1: Illustration of our framework.

on augmented sentences, contrastive learning is performed. Two augmented sentences are labeled as similar if they originate from the same original sentence. Two augmented sentences are labeled as dissimilar if they originate from different original sentences. Contrastive losses (Hadsell et al., 2006) are defined on these similar and dissimilar pairs. A text encoder is pretrained by minimizing the contrastive losses.

We assume the end task is text classification. Our framework consists of the following learning stages, which are performed end-to end. At the first stage, we train four translation models to perform data augmentation using back translation. Each translation pair in the training set is associated with a weight. At the second stage, on augmented sentences created by the translation models, we perform contrastive learning to train a text encoder. At the third stage, using the encoder trained at the second stage as regularization, we finetune a text classification model. At the fourth stage, the classification model trained at the third stage is evaluated on a validation classification dataset and the weights of translation pairs at the first stage are updated by minimizing the validation loss. The four stages are performed in a four-level optimization framework. Figure 1 illustrates our framework. Next, we describe the four stages in detail.

3.2 Stage I: Training Machine Translation Model for Sentence Augmentation

Figure 2 shows the workflow of data augmentation. For an input sentence x, we augment it using back-translation (Sennrich et al., 2016). In our experiments, the language of the classification data is English. We use an English-to-German machine translation (MT) model to translate x to y. Then we use a German-to-English MT model to translate y to x'. Then x' is regarded as

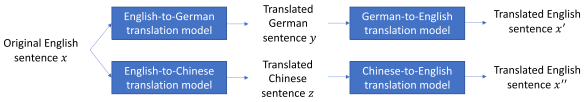

Figure 2: The workflow of data augmentation based on back translation.

an augmented sentence of x. Similarly, we use an English-to-Chinese MT model and a Chinese-to-English MT model to obtain another augmented sentence x''. We use German and Chinese as the two auxiliary languages because 1) both of them are resource-rich languages that have abundant translation data for model training; 2) they are sufficiently different from each other to achieve higher diversity in augmented examples.

To perform data augmentation based on back translation, we train four machine translation (MT) models: English-to-German, German-to-English, English-to-Chinese, and Chinese-to-English. Let W_{eg}, W_{ge}, W_{ec}, and W_{ce} denote these four MT models, and let $D_{eg} = \{d_i^{(eg)}\}_{i=1}^{N^{(eg)}}$, $D_{ge} = \{d_i^{(ge)}\}_{i=1}^{N^{(ge)}}$, $D_{ec} = \{d_i^{(ec)}\}_{i=1}^{N^{(ec)}}$, and $D_{ce} = \{d_i^{(ce)}\}_{i=1}^{N^{(ce)}}$ denote the corresponding datasets used to train these four models. Some translation data have a large domain discrepancy with the text classification data and should be excluded from training the translation models. Otherwise, the translation models trained using such out-of-domain translation data may not be able to generate meaningful augmentations for the text classification data due to the domain discrepancy. To identify and remove out-of-domain translation data, for $d_i^{(eg)}$, $d_i^{(ge)}$, $d_i^{(ec)}$, and $d_i^{(ce)}$, we associate them with weights $a_i^{(eg)}$, $a_i^{(ge)}$, $a_i^{(ec)}$, and $a_i^{(ce)}$, which are in $[0, 1]$. If the weight is close to 0, it means that the corresponding example has a large domain discrepancy with classification texts and should be excluded from training the MT models. These weights are used to reweight the training losses. At this stage, we solve the following optimization problems:

$$
\begin{aligned}
W_{eg}^*(A_{eg}) &= \operatorname*{argmin}_{W_{eg}} \sum_{i=1}^{N^{(eg)}} a_i^{(eg)} \ell_{mt}(d_i^{(eg)}; W_{eg}), \\
W_{ge}^*(A_{ge}) &= \operatorname*{argmin}_{W_{ge}} \sum_{i=1}^{N^{(ge)}} a_i^{(ge)} \ell_{mt}(d_i^{(ge)}; W_{ge}), \\
W_{ec}^*(A_{ec}) &= \operatorname*{argmin}_{W_{ec}} \sum_{i=1}^{N^{(ec)}} a_i^{(ec)} \ell_{mt}(d_i^{(ec)}; W_{ec}), \\
W_{ce}^*(A_{ce}) &= \operatorname*{argmin}_{W_{ce}} \sum_{i=1}^{N^{(ce)}} a_i^{(ce)} \ell_{mt}(d_i^{(ce)}; W_{ce}),
\end{aligned}
\tag{1}
$$

where $A_{eg} = \{a_i^{(eg)}\}_{i=1}^{N^{(eg)}}$, $A_{ge} = \{a_i^{(ge)}\}_{i=1}^{N^{(ge)}}$, $A_{ec} = \{a_i^{(ec)}\}_{i=1}^{N^{(ec)}}$, and $A_{ce} = \{a_i^{(ce)}\}_{i=1}^{N^{(ce)}}$. $\ell_{mt}(d_i^{(eg)}; W_{eg})$ is an MT loss defined on $d_i^{(eg)}$. If $a_i^{(eg)}$ is close to 0 (indicating $d_i^{(eg)}$ has large domain discrepancy with classification texts), this loss is made close to 0, effectively excluding $d_i^{(eg)}$ from training the MT model. These data weights are tentatively fixed at this stage and will be updated later. They cannot be updated at this stage by minimizing the training losses. Otherwise, trivial solutions will be yielded where all these weights are zero. Note that W_{eg}^* depends on A_{eg} since W_{eg}^* depends on $\sum_{i=1}^{N^{(eg)}} a_i^{(eg)} \ell_{mt}(d_i^{(eg)}; W_{eg})$ which is a function of A_{eg}.

Given an original English sentence x, we feed it into $W_{eg}^*(A_{eg})$ to get a translated German sentence, which is then fed into $W_{ge}^*(A_{ge})$ to get a translated English sentence x'. Meanwhile, we feed x into $W_{ec}^*(A_{ec})$ to get a translated Chinese sentence, which is then fed into $W_{ce}^*(A_{ce})$ to get another translated English sentence x''. x' and x'' are two augmented sentences of x. Since translation models are trained using in-domain translation examples that have large domain similarity with classification data, augmented examples generated by translation models are likely to be in the same domain as classification data as well; contrastive learning performed on these in-domain augmented examples is likely to produce latent representations that are suitable for representing classification data.

3.3 Stage II: Contrastive Learning

At the second stage, we perform CSSL pretraining. Given two augmented sentences, if they originate from the same original sentence, they are labeled as a positive pair; if they are from different sentences, they are labeled as a negative pair. Let x' and x'' denote two sentences augmented from the same original sentence x. Let $\{y_i\}_{i=1}^K$ denote K augmented sentences derived from original sentences different from x. Let U denote a text encoder such as BERT and $f(t;U)$ denote the embedding of a sentence t extracted by U. Let $s(r,t;U) = \exp(\text{sim}(f(r;U), f(t;U))/\tau)$ where $\text{sim}(\cdot,\cdot)$ denotes cosine similarity and τ is a temperature parameter. Let $\mathcal{A}$ denote $\{A_{eg}, A_{ge}, A_{ec}, A_{ce}\}$ and $\mathcal{W}^*(\mathcal{A})$ denote $\{W_{eg}^*(A_{eg}), W_{ge}^*(A_{ge}), W_{ec}^*(A_{ec}), W_{ce}^*(A_{ce})\}$. We define the following contrastive loss (Hadsell et al., 2006) on x:

$$\ell_c(x; U, \mathcal{W}^*(\mathcal{A})) = -\log \frac{s(x', x''; U)}{s(x', x''; U) + \sum_{i=1}^K s(x', y_i; U)}. \quad (2)$$

Note that x', x'', and $\{y_i\}_{i=1}^K$ are generated by $\mathcal{W}^*(\mathcal{A})$. At this stage, we solve the following optimization problem:

$$U^*(\mathcal{W}^*(\mathcal{A})) = \text{argmin}_U \sum\nolimits_x \ell_c(x; U, \mathcal{W}^*(\mathcal{A})). \quad (3)$$

3.4 Stage III: Finetuning Text Classifier

At the third stage, we finetune a text classification model where the text encoder is regularized by the encoder trained at the second stage. Let V and H denote the text encoder and classification head in the classification model. Let $D_{cls}^{(tr)}$ and $D_{cls}^{(val)}$ denote the training and validation sets of a classification dataset. The third stage amounts to solving the following problem:

$$V^*(U^*(\mathcal{W}^*(\mathcal{A}))), H^* = \underset{V,H}{\text{argmin}}\ L_{cls}(D_{cls}^{(tr)}; V, H) + \lambda \|V - U^*(\mathcal{W}^*(\mathcal{A}))\|_2^2, \quad (4)$$

where $L_{cls}(\cdot)$ is classification loss. $\|\cdot\|_2^2$ is an L2 regularizer that encourages the text encoder V to be close to the encoder $U^*(\mathcal{W}^*(\mathcal{A}))$ pretrained at the second stage. λ is a tradeoff parameter.

3.5 Stage IV: Update Weights of Translation Data

At the fourth stage, the classification model finetuned at the third stage is evaluated on the validation set and the weights of machine translation (MT) examples are updated by minimizing the validation loss:

$$\min_{\mathcal{A}} L_{cls}(D_{cls}^{(val)}; V^*(U^*(\mathcal{W}^*(\mathcal{A}))), H^*). \quad (5)$$

3.6 Four-Level Optimization Framework

Putting these pieces together, we have the following four-level optimization framework.

(Stage IV:)

$$\min_{\mathcal{A}} L_{cls}(D_{cls}^{(val)}; V^*(U^*(\mathcal{W}^*(\mathcal{A}))), H^*)$$

$s.t.$ (Stage III:)

$$V^*(U^*(\mathcal{W}^*(\mathcal{A}))), H^* = \operatorname*{argmin}_{V,H} L_{cls}(D_{cls}^{(tr)}; V, H) + \lambda \|V - U^*(\mathcal{W}^*(\mathcal{A}))\|_2^2$$

(Stage II:)

$$U^*(\mathcal{W}^*(\mathcal{A})) = \text{argmin}_U \sum\nolimits_x \ell_c(x; U, \mathcal{W}^*(\mathcal{A}))$$

(Stage I:)

$$W_{eg}^*(A_{eg}) = \operatorname*{argmin}_{W_{eg}} \sum\nolimits_{i=1}^{N^{(eg)}} a_i^{(eg)} \ell_{mt}(d_i^{(eg)}; W_{eg})$$

$$W_{ge}^*(A_{ge}) = \operatorname*{argmin}_{W_{ge}} \sum\nolimits_{i=1}^{N^{(ge)}} a_i^{(ge)} \ell_{mt}(d_i^{(ge)}; W_{ge})$$

$$W_{ec}^*(A_{ec}) = \operatorname*{argmin}_{W_{ec}} \sum\nolimits_{i=1}^{N^{(ec)}} a_i^{(ec)} \ell_{mt}(d_i^{(ec)}; W_{ec})$$

$$W_{ce}^*(A_{ce}) = \operatorname*{argmin}_{W_{ce}} \sum\nolimits_{i=1}^{N^{(ce)}} a_i^{(ce)} \ell_{mt}(d_i^{(ce)}; W_{ce}) \quad (6)$$

3.7 Reducing Memory and Computation Cost

In the proposed formulation in Eq. (6), there are four translation models and two text encoders. Storing these models and performing computation on them will incur a lot of memory and computation costs. In this section, we discuss how to reduce such costs, via parameter sharing (Sachan and Neubig, 2018). For the two encoders—one pretrained during contrastive learning and the other finetuned during text classification, we can let them share the same weight parameters. As such, the third stage becomes:

$$H^*(U^*(\mathcal{W}^*(\mathcal{A}))) = \operatorname*{argmin}_{H} L_{cls}(D_{cls}^{(tr)}; U^*(\mathcal{W}^*(\mathcal{A})), H), \quad (7)$$

and the fourth stage becomes:

$$\min_{\mathcal{A}} \; L_{cls}(D_{cls}^{(val)}; U^*(\mathcal{W}^*(\mathcal{A})), H^*(U^*(\mathcal{W}^*(\mathcal{A})))). \quad (8)$$

For the four translation models, they involve four encoders and four decoders, for three languages. For the same language, we let its encoders and decoders in the four translation models share the same parameters. By doing this, the eight encoders/decoders are reduced to three encoders/decoders.

3.8 Optimization Algorithm

We develop an optimization algorithm to solve the problem in Eq. (6). A similar algorithm can be developed for the memory/computation cost reduced framework in Section 3.7. Let $\nabla^2_{Y,X} f(X,Y)$ denote $\frac{\partial f(X,Y)}{\partial X \partial Y}$. Following Liu et al. (2018), at the first stage (where the training data are translation examples), we approximate $W_{eg}^*(A_{eg})$, $W_{ge}^*(A_{ge})$, $W_{ec}^*(A_{ec})$, and $W_{ce}^*(A_{ce})$ using one-step gradient descent updates of W_{eg}, W_{ge}, W_{ec}, and W_{ce}:

$$W_{eg}^*(A_{eg}) \approx W'_{eg} = W_{eg} - \eta_w \nabla_{W_{eg}} \sum\nolimits_{i=1}^{N^{(eg)}} a_i^{(eg)} \ell_{mt}(d_i^{(eg)}; W_{eg}), \quad (9)$$

$$W_{ge}^*(A_{ge}) \approx W'_{ge} = W_{ge} - \eta_w \nabla_{W_{ge}} \sum\nolimits_{i=1}^{N^{(ge)}} a_i^{(ge)} \ell_{mt}(d_i^{(ge)}; W_{ge}), \quad (10)$$

$$W_{ec}^*(A_{ec}) \approx W'_{ec} = W_{ec} - \eta_w \nabla_{W_{ec}} \sum\nolimits_{i=1}^{N^{(ec)}} a_i^{(ec)} \ell_{mt}(d_i^{(ec)}; W_{ec}), \quad (11)$$

$$W_{ce}^*(A_{ce}) \approx W'_{ce} = W_{ce} - \eta_w \nabla_{W_{ce}} \sum\nolimits_{i=1}^{N^{(ce)}} a_i^{(ce)} \ell_{mt}(d_i^{(ce)}; W_{ce}), \quad (12)$$

At the second stage, we use these approximate models (including W'_{eg}, W'_{ge}, W'_{ec}, W'_{ce}) to generate augmented sentences and define contrastive losses on these augmented sentences. Let $\mathcal{W}'$ denote $\{W'_{eg}, W'_{ge}, W'_{ec}, W'_{ce}\}$. The summation of all contrastive losses can be written as $\sum_x \ell_c(x; U, \mathcal{W}')$. Then we approximate $U^*(\mathcal{W}^*(\mathcal{A}))$ using one-step gradient descent update of U with respect to $\sum_x \ell_c(x; U, \mathcal{W}')$:

$$U^*(\mathcal{W}^*(\mathcal{A})) \approx U' = U - \eta_u \nabla_U \sum\nolimits_x \ell_c(x; U, \mathcal{W}'). \quad (13)$$

At the third stage (where the training data are classification examples), we plug the approximation $U^*(\mathcal{W}^*(\mathcal{A})) \approx U'$ into Eq. (4) and get an approximate objective. Then we approximate $V^*(U^*(\mathcal{W}^*(\mathcal{A})))$ and H^* using one-step gradient descent update of V and H with respect to the approximated objective:

$$V^*(U^*(\mathcal{W}^*(\mathcal{A}))) \approx V' = V - \eta_v \nabla_V (L_{cls}(D_{cls}^{(tr)}; V, H) + \lambda \|V - U'\|_2^2), \quad (14)$$

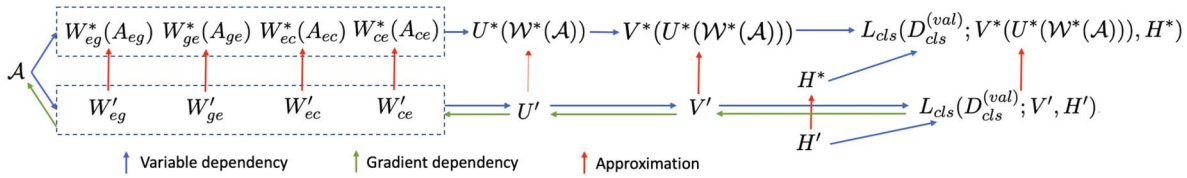

Figure 3: Dependency between variables and gradients.

Algorithm 1: Optimization algorithm.

while *not converged* **do**
1. Update MT models using Eq. (9) to Eq. (12)
2. Update text encoder U in contrastive SSL using Eq. (13)
3. Update text encoder V and classification head H in the classification model using Eq. (14) and Eq. (15)
4. Update machine translation example weights $\mathcal{A}$ using Eq. (16)

end

$$H^* \approx H' = H - \eta_h \nabla_H L_{cls}(D^{(tr)}_{cls}; V, H). \quad (15)$$

At the fourth stage (where the validation data are classification examples), we plug the approximations $V^*(U^*(\mathcal{W}^*(\mathcal{A}))) \approx V'$ and $H^* \approx H'$ into Eq. (5) and get an approximated validation loss, then update $\mathcal{A}$ by performing one-step gradient descent with respect to the approximated validation loss.

$$\mathcal{A} \leftarrow \mathcal{A} - \eta_a \nabla_{\mathcal{A}} L_{cls}(D^{(val)}_{cls}; V', H'). \quad (16)$$

After $\mathcal{A}$ is updated, W'_{eg}, W'_{ge}, W'_{ec}, W'_{ce} in Eqs. (9–12), which are functions of $\mathcal{A}$, need to be updated as well. Further, U' in Eq. (13), which is a function of W'_{eg}, W'_{ge}, W'_{ec}, W'_{ce}, needs to be updated as well. V' in Eq. (14), which is a function of U', needs to be updated. After V' is updated, $\mathcal{A}$ in Eq. (16), which is a function of V', needs to be updated again. A further updated $\mathcal{A}$ will render all other variables to be updated again. This update process iterates until convergence. In each iteration, we update each variable with one-step gradient descent, then move to updating the next variable. The iterative algorithm is summarized in Algorithm 1. Blue arrows in Figure 3 show the dependency between variables.

Next, we discuss how to calculate the gradient $\nabla_{\mathcal{A}} L_{cls}(D^{(val)}_{cls}; V', H')$ in Eq. (16). According to the chain rule, we have:

$$\nabla_{\mathcal{A}} L_{cls}(D^{(val)}_{cls}; V', H') = \frac{\partial \mathcal{W}'}{\partial \mathcal{A}} \frac{\partial U'}{\partial \mathcal{W}'} \frac{\partial V'}{\partial U'} \nabla_{V'} L_{cls}(D^{(val)}_{cls}; V', H'). \quad (17)$$

From Eq. (14), we have:

$$\frac{\partial V'}{\partial U'} = -\eta_v \lambda \nabla_{U',V} \|V - U'\|_2^2, \quad (18)$$

From Eq. (13), we have:

$$\frac{\partial U'}{\partial \mathcal{W}'} = -\eta_u \nabla_{W',U} \sum_x \ell_c(x; U, \mathcal{W}'), \quad (19)$$

From Eqs. (9–12), we have:

$$\frac{\partial W'_{eg}}{\partial A_{eg}} = -\eta_w \nabla^2_{A_{eg},W_{eg}} \sum_{i=1}^{N^{(eg)}} a_i^{(eg)} \ell_{mt}(d_i^{(eg)}; W_{eg}), \quad (20)$$

$$\frac{\partial W'_{ge}}{\partial A_{ge}} = -\eta_w \nabla^2_{A_{ge},W_{ge}} \sum_{i=1}^{N^{(ge)}} a_i^{(ge)} \ell_{mt}(d_i^{(ge)}; W_{ge}), \quad (21)$$

$$\frac{\partial W'_{ec}}{\partial A_{ec}} = -\eta_w \nabla^2_{A_{ec},W_{ec}} \sum_{i=1}^{N^{(ec)}} a_i^{(ec)} \ell_{mt}(d_i^{(ec)}; W_{ec}), \quad (22)$$

$$\frac{\partial W'_{ce}}{\partial A_{ce}} = -\eta_w \nabla^2_{A_{ce},W_{ce}} \sum_{i=1}^{N^{(ce)}} a_i^{(ce)} \ell_{mt}(d_i^{(ce)}; W_{ce}). \quad (23)$$

Green arrows in Figure 3 show the dependency during gradient calculation. The gradients in Eqs. (9–15) and Eqs. (18–23) can be automatically calculated using auto differentiation (e.g., autograd in PyTorch). The gradient in Eq. (17) needs to be manually implemented. When approximating optimal solutions at Stage I-III and updating $\mathcal{A}$ at Stage IV, we calculate stochastic

	CoLA	RTE	QNLI	STS-B	MRPC	WNLI	SST-2	MNLI (m/mm)	QQP	AX
Train	8551	2490	104743	5749	3668	635	67349	392702	363871	–
Dev	1043	277	5463	1500	408	71	872	9815/9832	40432	–
Test	1063	3000	5463	1379	1725	146	1821	9796/9847	390965	1104

Table 1: Split statistics of GLUE datasets.

gradients on mini-batches instead of full gradients on the entire dataset.

4 Experiments

In this section, we evaluate our framework on eleven English understanding tasks in the GLUE (Wang et al., 2018b) benchmark and eight text datasets used in Gururangan et al. (2020)

4.1 Tasks and Datasets

For text classification, we use two collections of datasets. The first collection is from the General Language Understanding Evaluation (GLUE) benchmark, which has 11 tasks, including 2 single-sentence tasks including CoLA (Warstadt et al., 2019) and SST-2 (Socher et al., 2013), 3 similarity and paraphrase tasks including MRPC (Dolan and Brockett, 2005), QQP,[1] and STS-B (Cer et al., 2017), and 5 inference tasks including MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), RTE (Dagan et al., 2005), and WNLI (Levesque et al., 2012). Table 1 shows the split statistics of GLUE datasets. The second collection is from Gururangan et al. (2020), including CHEMPROT (Kringelum et al., 2016), RCT (Dernoncourt and Lee, 2017), ACL-ARC (Jurgens et al., 2018), SCIERC (Luan et al., 2018), HYPERPARTISAN (Kiesel et al., 2019), AGNEWS (Zhang et al., 2015), HELPFULNESS (McAuley et al., 2015), and IMDB (Maas et al., 2011). In our method, we split the original training set into a new training set and a validation set, with a ratio of 1:1. The new training set is used as $D_{cls}^{(tr)}$ and the validation set is used as $D_{cls}^{(val)}$.

For machine translation, we use 3K English-Chinese and 3K English-German language pairs randomly sampled from WMT17.[2] For contrastive learning, it is performed on all input texts (excluding labels) of training datasets in the 11 GLUE tasks.

[1] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs.

[2] http://www.statmt.org/wmt17/translation-task.html.

4.2 Experimental Settings

For translation models, we use those experimented in Britz et al. (2017), which are encoder-decoder models with attention. The encoder and decoder are both 4-layer bi-directional LSTM networks with a hidden size of 512. The attentions are additive, with a dimension of 512. For classifiers, BERT is used for GLUE and RoBERTa is used for datasets in Gururangan et al. (2020). The gumbel-softmax trick (Jang et al., 2017; Maddison et al., 2017) is leveraged to deal with the non-differentiability of words.

We use MoCo (He et al., 2020) to implement the contrastive learning method. Text encoders are initialized using pretrained BERT (Devlin et al., 2019) or pretrained RoBERTa (Liu et al., 2019). In MoCo, the size of the queue (which is the hyperparameter K in Section 3.3) was set to 96606. The coefficient of MoCo momentum of updating the key encoder was set to 0.999. The temperature parameter (which is the hyperparameter τ in Section 3.3) in the contrastive loss was set to 0.07. A multi-layer perceptron head was used. For MoCo training, a stochastic gradient descent solver with momentum was used. Mini-batch size was set to 16. Initial learning rate was set to $4 \cdot 10^{-5}$. Learning rate was adjusted using cosine scheduling. Weight decay was used with a coefficient of $1 \cdot 10^{-5}$.

For classification on GLUE tasks, the classification head is set to a linear layer. The maximum sequence length was set to 128. The tradeoff parameter λ in Eq. (4) is set to 0.1. Minibatch size was set to 16. The learning rate was set to $3 \cdot 10^{-5}$ for CoLA, MNLI, STS-B; $2 \cdot 10^{-5}$ for RTE, QNLI, MRPC, SST-2, WNLI; and $1 \cdot 10^{-5}$ for QQP. The number of training epochs was set to 100.

Hyperparameter Tuning Details For most hyperparameters in MoCo and LSTM, we use the default values given in He et al. (2020) and Britz et al. (2017). The tradeoff parameter λ is tuned in $\{0.01, 0.05, 0.1, 0.5, 1\}$ on the development set. For each configuration of λ, we run our method on $D_{cls}^{(tr)}$ (one half of the training set) and

$D_{cls}^{(val)}$ (the other half of the training set). Then we measure the performance of the trained model on the development set. The λ value yielding the best performance is selected. We tuned the hyperparameters of baselines extensively, where the tuning time for each baseline is roughly the same as that for our method.

4.3 Baselines

We compare our methods with the following baselines. Let Ours-SPS denote the proposed framework in Eq. (6) which performs soft parameter-sharing (SPS) between V and U via regularization, and let Ours-HPS denote the framework in Section 3.7 which performs hard parameter-sharing (HPS) where V and U are the same.

- **Vanilla RoBERTa** (Liu et al., 2019). The Transformer-based encoder is initialized with pretrained RoBERTa. A text classification model is formed by stacking the pretrained encoder and a classification head, with an architecture that is the same as that in Liu et al. (2019). The classification head is a feedforward layer, where the nonlinear activation function is tanh. Learned encoding of the special token `[CLS]` is fed into the classification head to predict the class label. Then we finetune the classification model on a classification dataset.

- **Vanilla BERT** (Devlin et al., 2019). This approach is similar to vanilla RoBERTa. The only difference is that the Transformer-based encoder is initialized by pretrained BERT (Devlin et al., 2019) instead of RoBERTa.

- **TAPT: Task Adaptive Pretraining** (Gururangan et al., 2020). In this approach, given a target dataset D_t, the pretrained BERT or RoBERTa on external data is further pretrained on the input sentences in D_t by predicting masked tokens.

- **SimCSE** (Gao et al., 2021). In this approach, the same input sentence is fed into a pretrained RoBERTa encoder twice by applying different dropout masks, to get two different embeddings. These two embeddings are labeled as being ‘‘similar’’. Embeddings of different sentences are labeled as being ‘‘dissimilar’’. Contrastive learning is performed on these ‘‘similar’’ and ‘‘dissimilar’’ pairs.

- **CSSL-Separate**. In this approach, data augmentation, contrastive learning, and text classification are performed separately. We first train machine translation models and use them to perform sentence augmentation. Then on augmented sentences, we perform contrastive learning. Finally, using the text encoder pretrained by contrastive learning as initialization, we finetune the classification model. When performing contrastive learning, the text encoder is initialized using pretrained RoBERTa or BERT.

- **CSSL-MTL**. This approach is similar to CSSL-Separate, except that the CSSL task and classification task are performed jointly in a multi-task learning (MTL) framework, by minimizing the weighted sum of their losses. The weight is 0.01 for CSSL loss and is 1 for classification loss.

4.4 Results

4.4.1 Results in BERT-Based Experiments

In BERT-based experiments, the text encoder is initialized using BERT, before contrastive learning is performed. Tables 2 and 3 show the results on GLUE test sets, in BERT-based experiments. Our methods including Ours-SPS and Ours-HPS outperform all baselines on average scores. Out of the 11 tasks (MNLI-m and MNLI-mm are treated as two separate tasks), Ours-SPS outperforms all baselines on 8 tasks; Ours-HPS outperforms all baselines on 7 tasks. These results demonstrate the effectiveness of our end-to-end frameworks. Via parameter sharing, Ours-HPS has much smaller memory and computation costs than Ours-SPS, with a small sacrifice of classification performance. The inference costs of our methods are similar to those of baselines. During inference, only V and H are needed, which are the same as baseline models. Figure 4 shows the accuracy curve of Ours-HPS on the RTE validation set ($D_{cls}^{(val)}$) under different runs. As can be seen, our algorithm converges well.

We present the following analysis. **First**, the reason that our methods outperform CSSL-Separate and CSSL-MTL is that in our methods, data augmentation and contrastive learning are performed end-to-end where the training of translation models (used for data augmentation)

	Train Time (hours)	Memory (GB)	Train Data (millions)	Parameters (millions)	CoLA (Matthew)	SST-2 (Acc.)	RTE (Acc.)	QNLI (Acc.)	MRPC (Acc./F1)
BERT	6.3	11.7	1.019	345	60.5	94.9	70.1	92.7	85.4/89.3
TAPT	13.5	11.8	1.019	345	61.3	94.4	70.3	92.4	85.9/89.5
SimCSE	16.4	12.1	1.019	690	59.5	94.3	71.2	**92.9**	85.9/89.8
CSSL-Separate	16.1	12.0	1.025	713	59.4	94.5	71.4	92.8	85.8/89.6
CSSL-MTL	16.9	12.2	1.025	713	59.7	94.7	71.2	92.5	86.0/89.6
Ours-SPS	28.2	20.5	1.025	713	**63.0**	**95.8**	**72.5**	**93.2**	**86.1/89.9**
Ours-HPS	17.4	12.7	1.025	356	**62.4**	**95.3**	**72.4**	92.5	**86.3/89.9**

Table 2: Results on GLUE test sets, using BERT for model initialization. The results are obtained from the GLUE evaluation server. The best results are bolded. The second best results are bolded and underlined. Models evaluated on AX are trained on the training dataset of MNLI. Matthew denotes Matthew correlation. Acc. denotes accuracy.

	MNLI-m/mm (Accuracy)	QQP (Accuracy/F1)	STS-B (Pearson/ Spearman)	WNLI (Accuracy)	AX (Matthew)	Average
BERT	86.7/85.9	89.3/72.1	87.6/86.5	65.1	39.6	80.5
TAPT	85.7/84.4	89.6/71.9	88.1/87.0	65.8	39.3	80.6
SimCSE	87.1/86.4	90.5/72.5	87.8/86.9	65.8	39.6	80.8
CSSL-Separate	**87.3/86.6**	**90.6**/72.7	87.4/86.6	65.5	39.6	80.8
CSSL-MTL	**87.4/86.8**	**90.9/72.9**	87.3/86.6	65.4	39.6	80.8
Ours-SPS	86.7/86.2	90.0/**72.9**	**88.2/87.3**	**66.9**	**40.3**	**81.7**
Ours-HPS	86.8/86.2	89.8/72.8	**88.3/87.3**	**66.1**	**40.2**	**81.4**

Table 3: Continuation of Table 2. Pearson, Spearman, and Matthew denote the corresponding correlations.

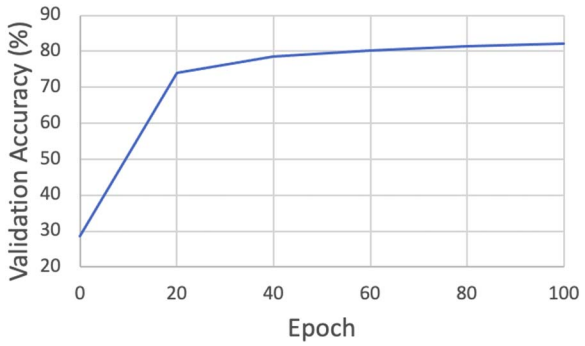

Figure 4: Accuracy on RTE validation set ($D_{cls}^{(val)}$).

is guided by the contrastive learning performance and the augmented sentences are encouraged to be suitable for performing the contrastive learning task. In contrast, in CSSL-Separate and CSSL-MTL, data augmentation and contrastive learning are performed separately. Consequently, the augmented data may not be optimal for performing contrastive learning. **Second**, the reason that Ours-SPS outperforms Ours-HPS is that in Ours-SPS, while the classification model is regularized by the CSSL-pretrained text encoder, they are not exactly the same. This gives the classification model some flexibility in capturing the unique properties of classification data. In contrast, in Ours-HPS, the classification model and the CSSL-pretrained text encoder are required to be exactly the same, which might be too restrictive. On the other hand, it is worth noting that the classification performance gap between Ours-HPS and CSSL-pretrained is not very large while the memory and computation cost of Ours-HPS are much smaller.

Third, overall, CSSL-MTL works better than CSSL-Separate. Out of the 11 tasks, CSSL-MTL outperforms CSSL-Separate on 6 tasks and is on par with CSSL-Separate on 1 task. The reason is that in CSSL-MTL, contrastive learning and classification are performed jointly, which enables these two tasks to mutually benefit from each other. In contrast, in CSSL-Separate, contrastive learning and classification are performed separately. While contrastive learning influences classification, classification does not provide any feedback to contrastive learning. **Fourth**, CSSL-Separate and SimCSE are in general on par with each other. The only difference between these two methods is that CSSL-Separate uses back-translation for data augmentation while SimCSE uses dropout masks. This shows that these two

	CoLA (Matthew Corr.)	SST-2 (Accuracy)	RTE (Accuracy)	QNLI (Accuracy)	MRPC (Accuracy/F1)
BERT (from Lan et al.)	60.6	93.2	70.4	92.3	88.0/–
BERT (our run)	62.1	93.1	74.0	92.1	86.8/90.8
TAPT	61.2	93.1	74.0	92.0	85.3/89.8
SimCSE	62.0	93.5	73.5	92.2	86.8/**90.9**
CSSL-Separate	62.3	93.7	72.0	92.5	87.0/**90.9**
CSSL-MTL	62.7	93.7	72.6	92.3	87.1/**90.9**
No-CL	62.3	93.2	74.0	92.2	86.9/90.8
No-FT	62.4	93.6	72.3	92.4	87.0/**90.9**
No-BT	62.8	93.7	74.0	92.5	87.1/**90.9**
Domain-Reweight	62.7	93.8	72.5	92.6	87.1/**90.9**
Fix-Weight-Separate	62.7	93.8	72.8	92.6	87.1/**90.9**
Fix-Weight-MTL	62.9	93.8	73.2	92.4	87.2/**90.9**
DANN	62.4	93.7	72.2	92.4	87.1/**90.9**
CAN	62.3	93.6	72.3	92.4	87.2/**90.9**
Ours-Transformer-MT	63.2	93.9	74.2	92.6	87.2/**90.9**
Ours-SPS	**63.6**	**94.0**	**74.8**	**92.9**	**87.8/91.1**
Ours-HPS	**63.3**	**93.9**	**74.4**	**92.7**	**87.3/90.9**

Table 4: Results on GLUE development sets, using BERT for model initialization. The result is the median of five runs. Due to randomness of model initialization, our median performance is not the same as that reported in Lan et al. (2019).

augmentation methods work equally well for contrastive learning on texts. **Fifth**, CSSL-Separate and SimCSE outperform TAPT. In TAPT, the self-supervised task is masked token prediction. This shows that contrastive learning is a more effective self-supervised learning method than masked token prediction. **Sixth**, CSSL-Separate and SimCSE perform better than BERT, which further demonstrates the effectiveness of contrastive learning. **Seventh**, the improvement achieved by our methods is more prominent on smaller datasets such as WNLI and RTE. This is because, for smaller datasets, it is more necessary to leverage unlabeled data via contrastive learning to learn overfitting-resilient representations. **Eighth**, for tasks that have similar data size, the improvement of our method is more prominent on similarity tasks than on other types of tasks. For example, QQP and MNLI have similar data size; our method achieves better improvement on QQP, which is a similarity task, than on MNLI, which is an inference task.

Tables 4 and 5 show the results of BERT-based experiments on the development sets of GLUE. Our methods outperform all baselines on different tasks. The analysis of reasons is similar to that for Tables 2 and 3. Note that our method can be applied to other text encoders besides BERT. For example, our method can be applied to Turing-NLR-v5 (Bajaj et al., 2022). This encoder achieves state-of-the-art performance on the GLUE leaderboard, with an average score of 91.2.

4.4.2 Results in RoBERTa-Based Experiments

In RoBERTa-based experiments, text encoders are initialized using RoBERTa, before contrastive learning is performed. These experiments are conducted on datasets used in Gururangan et al. (2020). Table 6 shows the results. Our methods outperform all baselines. The analysis of reasons is similar to that for results in Tables 2 and 3.

4.4.3 Ablation Studies

To verify whether the individual components in our method are necessary, we perform the following ablation studies.

- No contrastive learning (**No-CL**). Contrastive learning is removed from the framework. For each text-label pair $(x, y) \in D_{cls}^{(tr)}$, the input text x is fed into the four translation models to generate two augmentations x' and x''. Then (x', y) and (x'', y) are utilized as augmented data to train the classification model directly.

	MNLI-m/mm (Accuracy)	QQP (Accuracy/F1)	STS-B (Pearson Corr./ Spearman Corr.)	WNLI (Accuracy)
BERT (from Lan et al.)	86.6/–	91.3/–	90.0/–	–
BERT (our run)	86.2/86.0	91.3/88.3	90.4/90.0	**56.3**
TAPT	85.6/85.5	91.5/88.7	90.6/90.2	**53.5**
SimCSE	86.4/86.1	91.2/88.1	90.5/90.1	**56.3**
CSSL-Separate	86.6/86.4	91.4/88.5	90.2/89.9	**56.3**
CSSL-MTL	86.6/86.5	91.5/88.6	90.4/90.1	**56.3**
No-CL	86.3/86.1	91.3/88.4	90.5/90.2	**56.3**
No-FT	86.6/86.5	91.4/88.4	90.4/90.0	**56.3**
No-BT	86.7/86.6	91.5/88.7	90.7/90.3	**56.3**
Domain-Reweight	86.6/86.5	91.5/88.6	90.4/90.0	**56.3**
Fix-Weight-Separate	86.7/86.6	91.5/88.7	90.4/90.0	**56.3**
Fix-Weight-MTL	86.7/86.6	91.6/88.8	90.5/90.2	**56.3**
DANN	86.6/86.4	91.5/88.5	90.3/90.1	**56.3**
CAN	86.6/86.5	91.6/88.5	90.2/90.1	**56.3**
Ours-Transformer-MT	**86.7**/86.8	91.6/88.9	90.7/90.3	**56.3**
Ours-SPS	**86.9/87.0**	**91.9/89.2**	**91.0/90.8**	56.3
Ours-HPS	**86.9/86.9**	**91.7/89.1**	**90.8/90.5**	**56.3**

Table 5: Continuation of Table 4.

Dataset	RoBERTa	TAPT	SimCSE	CSSL-Separate	CSSL-MTL	Ours-SPS	Ours-HPS
ChemProt	$81.9_{1.0}$	$82.6_{0.4}$	$83.2_{0.2}$	$82.9_{0.5}$	$83.4_{0.3}$	$\mathbf{84.5}_{0.4}$	$\mathbf{84.2}_{0.3}$
	$87.2_{0.1}$	$87.7_{0.1}$	$87.6_{0.1}$	$87.6_{0.1}$	$87.7_{0.1}$	$\mathbf{88.0}_{0.1}$	$\mathbf{87.9}_{0.1}$
ACL-ARC	$63.0_{5.8}$	$67.4_{1.8}$	$69.5_{2.6}$	$68.7_{3.1}$	$70.8_{3.7}$	$\mathbf{75.6}_{2.9}$	$\mathbf{75.3}_{1.5}$
SciERC	$77.3_{1.9}$	$79.3_{1.5}$	$80.5_{0.7}$	$80.9_{1.3}$	$81.2_{0.9}$	$\mathbf{82.1}_{1.1}$	$\mathbf{81.9}_{0.6}$
HyperPartisan	$86.6_{0.9}$	$90.4_{5.2}$	$90.9_{2.7}$	$91.4_{3.3}$	$90.7_{1.9}$	$\mathbf{92.3}_{2.1}$	$\mathbf{91.8}_{1.6}$
AGNews	$93.9_{0.2}$	$94.5_{0.1}$	$94.7_{0.1}$	$94.3_{0.1}$	$94.5_{0.1}$	$\mathbf{95.1}_{0.1}$	$\mathbf{94.9}_{0.1}$
Helpfulness	$65.1_{3.4}$	$68.5_{1.9}$	$68.7_{1.7}$	$69.2_{0.5}$	$69.5_{1.1}$	$\mathbf{70.9}_{0.2}$	$\mathbf{70.4}_{0.4}$
IMDB	$95.0_{0.2}$	$95.5_{0.1}$	$95.7_{0.1}$	$95.6_{0.1}$	$95.3_{0.1}$	$\mathbf{96.0}_{0.1}$	$\mathbf{95.8}_{0.1}$

Table 6: Results of RoBERTa-based experiments on datasets used in Gururangan et al. (2020). The results of vanilla RoBERTa and TAPT are taken from Gururangan et al. (2020). Each method runs four times with random initialization. In the x_y formatted results, x denotes average and y denotes standard deviation. Following Gururangan et al. (2020), the results on ChemProt and RCT are micro-F1; the results on other datasets are macro-F1. The best results are bolded. The second best results are bolded and underlined.

- No finetuning (**No-FT**). At the third stage, the encoder V in the classification model is set to $U^*(\mathcal{W}^*(\mathcal{A}))$ without being finetuned.
- Replacing back-translation with SimCSE (**No-BT**). Instead of learning machine translation models for text augmentation, we use the augmentation method proposed in SimCSE (Gao et al., 2021) for augmentation.
- **Domain-Reweight**. In CSSL-Separate, we reweight self-supervised training examples based on their domain relatedness to classification data, and perform contrastive learning on reweighted examples. Domain relatedness is calculated using an $\mathcal{H}$-divergence based metric (Elsahar and Gallé, 2019).
- **Fix-Weight-Separate and Fix-Weight-MTL**: Learn translation sample weights using our method, fix them, then run CSSL-Separate and CSSL-MTL on reweighted translation examples.

- Compare with other domain adaptation methods, including **DANN** (Ganin et al., 2016) and **CAN** (Kang et al., 2019). We use these methods to align the domains of input sentences in the translation and classification datasets.
- **Ours-Transformer-MT**: In Ours-HPS, using Transformer (specifically, pretrained BERT) for machine translation, instead of using attentional LSTM.

Tables 4 and 5 show results on GLUE development sets, using BERT for model initialization. We make the following observations. **First**, our full methods work better than No-CL, No-FT, and No-BT. In these three ablation baselines, one component (which is contrastive learning, finetuning classifier, back translation, respectively) is removed from our full methods, yielding simpler methods. These results show that each component is useful and should not be removed, and simpler methods do not perform comparably well. **Second**, our methods work better than Domain-Reweight. The reason is that our methods perform reweighting of translation data together with performing other tasks (including training translation models, contrastive learning, finetuning, and validation), in an end-to-end manner. In this way, weights of translation data are influenced by other tasks and learned towards maximizing the classification performance. In contrast, Domain-Reweight performs reweighting of self-supervised training examples separately from other tasks. Weights calculated in this way are not guaranteed to be optimal for maximizing classification performance. On the other hand, Domain-Reweight outperforms CSSL-Separate, which shows that it is beneficial to reweight self-supervised training examples based on their domain relatedness to classification data. **Third**, our methods work better than Fix-Weight-Separate and Fix-Weight-MTL. The reason is that our methods generate augmented sentences and perform CL end-to-end while Fix-Weight-Separate and Fix-Weight-MTL perform these two tasks separately. In our end-to-end framework, guided by the performance of CL, the training of translation models is dynamically changing to generate augmented sentences that are better for improving CL. On the contrary, in Fix-Weight-Separate and Fix-Weight-MTL, the generation of augmented examples is not influenced by the CL task. Consequently, the generated augmentations may not be optimal for CL.

Method	**E→G**	**G→E**	**E→C**	**C→E**
MTL	14.8	15.3	14.2	14.6
Ours	**16.1**	**16.7**	**15.5**	**16.0**

Table 7: BLEU scores on test sets. E, G, C denote English, German, Chinese, respectively. $E \rightarrow G$ denotes English-to-German translation.

On the other hand, Fix-Weight-Separate and Fix-Weight-MTL outperform CSSL-Separate and CSSL-MTL, which further demonstrates the benefits of reweighting translation examples based on their domain similarity to classification data and the weights learned by our methods can accurately reflect domain similarity. **Fourth**, our methods work better than the two domain adaptation methods DANN and CAN. The reason is because many translation examples have large domain discrepancies with classification texts; it is difficult to adapt these translation examples into the domain of classification data. Our methods learn to remove such examples instead of forcefully adapting them. **Fifth**, comparing Ours-Transformer-MT (using BERT for translation) and Ours-HPS (using LSTM), we can see that BERT works slightly worse than LSTM. While BERT is more expressive, it has more weight parameters to learn than LSTM, which incurs higher risk of overfitting.

We also check whether our framework can improve machine translation (MT). Since the end goal of our work is improving text classification, we evaluate MT performance on selected translation examples that have large domain similarity to classification data. Domain similarity is calculated using $\mathcal{H}$-divergence (Elsahar and Gallé, 2019). Translation examples whose normalized $\mathcal{H}$-divergence is smaller than 0.5 are selected. MT models are trained on selected training examples and evaluated on selected test examples. Table 7 compares the BLUE (Papineni et al., 2002) scores (on test sets) of the four MT models trained in our framework and those trained via MTL (minimizing the sum of training losses of the four models) without using our framework. As can be seen, the models trained in our framework perform better. The reason is that our framework trains the translation models to generate linguistically meaningful augmented texts; by doing this, the translation models are encouraged to generate translations with higher linguistic quality.

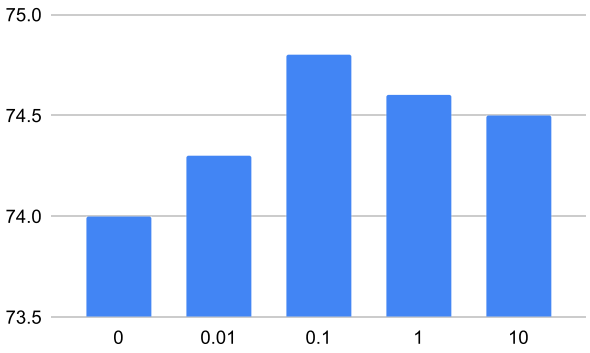

Figure 5: How the accuracy of the RTE development set changes with λ.

4.4.4 Parameter Sensitivity

Figure 5 shows how the classification accuracy on the development set of RTE changes with the tradeoff parameter λ of Ours-SPS. As can be seen, when λ increases from 0 to 0.1, the accuracy increases. This is because a larger λ encourages more knowledge transfer from the CSSL-pretrained encoder to the classification model. The representations learned in the contrastive SSL task help the classification model to learn. However, as we continue to increase λ, the accuracy decreases. This is because the classification model is too much biased to the CSSL-pretrained encoder and is less tailored to the classification data.

4.4.5 Qualitative Results

Table 8 shows some randomly sampled translation examples where the learned importance weights (a_i in Eq. (6)) are close to 0 or 1, when the classification task is SST-2 (the percentage of data gets near-zero weights is 35.2%). Due to space limitations, we only show the English sentences in translation pairs. As can be seen, translation sentences with near-zero weights have a large domain discrepancy with the SST-2 data. SST-2 mainly contains movie reviews while these zero-weight sentences are mainly about politics. Due to this domain discrepancy, these translation data is not suitable to train data augmentation models for SST-2. Our framework can effectively identify such out-of-domain translation data and exclude them from the training process. This is another reason that our end-to-end framework achieves better performance than baselines which lack the mechanism of removing out-of-domain translation data. On the other hand, in Table 8, sentences whose weights are close to one are more relevant to movie reviews.

Sentence	Weight
The government recently called for more balanced development, even proposing a ‘‘green index’’ to measure growth.	0
President-elect Donald Trump’s campaign narrative was based on the assumption that the US has fallen from its former greatness.	0
Russia considers the agreements from the 1990s unjust, based as they were on its weakness at the time, and it wants to revise them.	0
Publicity for the new film claims that it is ‘‘the first live-action film in the history of movies to star, and be told from the point of view of, a sentient animal.’’	1
Gore told the world in his Academy Award-winning movie (recently labeled ‘‘one-sided’’ and containing ‘‘scientific errors’’ by a British judge) to expect 20-foot sea-level rises over this century.	1
Jia’s movie is episodic; four loosely linked stories about lone acts of extreme violence, mostly culled from contemporary newspaper stories.	1

Table 8: Some randomly sampled translation examples with importance weights close to 0 and 1.

5 Conclusions, Discussion, and Future Work

In this paper, we propose an end-to-end framework for learning language representations based on contrastive learning. Different from existing contrastive learning methods that perform data augmentation and contrastive learning separately and thus cannot guarantee that the augmented data is optimal for contrastive learning, our method performs data augmentation and contrastive learning end-to-end in a unified framework so that data augmentation models are specifically trained for being suitable for contrastive learning. Our framework consists of four learning stages: 1) training machine translation models for text augmentation; 2) contrastive learning; 3) training a classification model; 4) updating weights of translation data by minimizing the validation loss of the classification model. We evaluate our framework on 11 English understanding tasks in the GLUE benchmark and 8 datasets in Gururangan et al. (2020). On both test set and development set, the experimental results demonstrate the effectiveness of our method.

One major limitation of our method is that it has larger computational and memory costs, due to the extra overhead of solving a four-level optimization based problem and storing MT models.

To reduce these costs, in addition to tying parameters, we will explore other techniques in future, such as reducing the update frequencies of MT models and MT data weights, applying diversity-promoting regularizers (Xie et al., 2017) to speed up convergence, performing core-set based mini-batch selection (Sinha et al., 2020) to speed up convergence, and so forth.

For future work, we plan to study more challenging loss functions for self-supervised learning. We are interested in investigating a ranking-based loss, where each sentence is augmented with a ranked list of sentences that have decreasing discrepancy with the original sentence. The auxiliary task is to predict the order given the augmented sentences. Predicting an order is presumably more challenging than binary classification (as adopted in existing contrastive SSL methods) and may facilitate the learning of better representations.

References

Payal Bajaj, Chenyan Xiong, Guolin Ke, Xiaodong Liu, Di He, Saurabh Tiwary, Tie-Yan Liu, Paul Bennett, Xia Song, and Jianfeng Gao. 2022. Metro: Efficient denoising pretraining of large scale autoencoding language models with model generated signals. *arXiv preprint arXiv:2204.06644*.

Atılım Güneş Baydin, Robert Cornish, David Martínez Rubio, Mark Schmidt, and Frank Wood. 2018. Online learning rate adaptation with hypergradient descent. In *Sixth International Conference on Learning Representations (ICLR), Vancouver, Canada, April 30 – May 3, 2018*.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. *Advances in Neural Information Processing Systems*, 13.

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1451, Copenhagen, Denmark. Association for Computational Linguistics.

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, Vancouver, Canada. Association for Computational Linguistics.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The Pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Stephan Dempe. 2002. *Foundations of Bilevel Programming*. Springer Science & Business Media.

Franck Dernoncourt and Ji Young Lee. 2017. PubMed 200k RCT: A dataset for sequential sentence classification in medical abstracts. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 308–313, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Hady Elsahar and Matthias Gallé. 2019. To annotate or not? Predicting performance drop under domain shift. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2163–2173. https://doi.org/10.18653/v1/D19-1222

Matthias Feurer, Jost Springenberg, and Frank Hutter. 2015. Initializing Bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29. https://doi.org/10.1609/aaai.v29i1.9354

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

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, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of ACL*. https://doi.org/10.18653/v1/2020.acl-main.740

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.

Olivier Henaff. 2020. Data-efficient image recognition with contrastive predictive coding. In *International Conference on Machine Learning*, pages 4182–4192. PMLR.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. *ICLR*.

David Jurgens, Srijan Kumar, Raine Hoover, Daniel A. McFarland, and Dan Jurafsky. 2018. Measuring the evolution of a scientific field through citation frames. *TACL*.

Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G. Hauptmann. 2019. Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4893–4902. https://doi.org/10.1109/CVPR.2019.00503

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan news detection. In *SemEval*. https://doi.org/10.18653/v1/S19-2145

Tassilo Klein and Moin Nabi. 2020. Contrastive self-supervised learning for commonsense reasoning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7517–7523.

Jens Kringelum, Sonny Kim Kjærulff, Søren Brunak, Ole Lund, Tudor I. Oprea, and Olivier Taboureau. 2016. ChemProt-3.0: A global chemical biology diseases mapping. In *Database*. https://doi.org/10.1093/database/bav123, PubMed: 26876982

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. AlBERT: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. 2020. CURL: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of

Proceedings of Machine Learning Research, pages 5639–5650. PMLR.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The Winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. In *International Conference on Learning Representations*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1360

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *ACL*.

C. Maddison, A. Mnih, and Y. Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the International Conference on Learning Representations*. International Conference on Learning Representations.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *ACM SIGIR*. https://doi.org/10.1145/2766462.2767755

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Lin Pan, Chung-Wei Hang, Avirup Sil, and Saloni Potdar. 2022. Improved text classification via contrastive adversarial training. *AAAI*.

Xiao Pan, Mingxuan Wang, Liwei Wu, and Lei Li. 2021. Contrastive learning for many-to-many multilingual neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 244–258. https://doi.org/10.18653/v1/2021.acl-long.21

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. Association for Computational Linguistics. https://doi.org/10.3115/1073083.1073135

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. https://doi.org/10.18653/v1/D16-1264

Zhongzheng Ren, Raymond Yeh, and Alexander Schwing. 2020. Not all unlabeled data are equal: Learning to weight data in semi-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21786–21797. Curran Associates, Inc.

Devendra Sachan and Graham Neubig. 2018. Parameter sharing methods for multilingual

self-attentional translation models. In *Conference on Machine Translation*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96. https://doi.org/10.18653/v1/P16-1009

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. 2019. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, pages 1919–1930.

Samarth Sinha, Han Zhang, Anirudh Goyal, Yoshua Bengio, Hugo Larochelle, and Augustus Odena. 2020. Small-gan: Speeding up gan training using core-sets. In *International Conference on Machine Learning*, pages 9005–9015. PMLR.

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.

Peng Su, Yifan Peng, and K. Vijay-Shanker. 2021. Improving bert model using contrastive learning for biomedical relation extraction. In *Proceedings of the 20th Workshop on Biomedical Language Processing*, pages 1–10.

Felipe Petroski Such, Aditya Rawal, Joel Lehman, Kenneth Stanley, and Jeffrey Clune. 2020. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. In *International Conference on Machine Learning*, pages 9206–9216. PMLR.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv: 1804.07461*. https://doi.org/10.18653/v1/W18-5446

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*. https://doi.org/10.18653/v1/W18-5446

Dong Wang, Ning Ding, Piji Li, and Haitao Zheng. 2021. Cline: Contrastive learning with semantic negative examples for natural language understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2332–2342. https://doi.org/10.18653/v1/2021.acl-long.181

Yulin Wang, Jiayi Guo, Shiji Song, and Gao Huang. 2020. Meta-semi: A meta-learning approach for semi-supervised learning. *CoRR*, abs/2007.02394.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641. https://doi.org/10.1162/tacl_a_00290

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. https://doi.org/10.18653/v1/N18-1101

Pengtao Xie, Aarti Singh, and Eric P. Xing. 2017. Uncorrelation and evenness: A new diversity-promoting regularizer. In *International Conference on Machine Learning*, pages 3811–3820. PMLR.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NeurIPS*.

Guoqing Zheng, Ahmed Hassan Awadallah, and Susan Dumais. 2021. Meta label correction for learning with weak supervision. *AAAI*. https://doi.org/10.1609/aaai.v35i12.17319