Soft Representation Learning for Sparse Transfer

Haeju Park[1] **Jinyoung Yeo**[2] **Gengyu Wang**[1] **Seung-won Hwang**[1]
Department of Computer Science, Yonsei University, Seoul, Korea[1]
T-Brain, AI Center, SK Telecom, Seoul, Korea[2]
{phj0225, posuer, seungwonh}@yonsei.ac.kr
bradyeo33@sktbrain.com

Abstract

Transfer learning is effective for improving the performance of tasks that are related, and Multi-task learning (MTL) and Cross-lingual learning (CLL) are important instances. This paper argues that hard-parameter sharing, of hard-coding layers shared across different tasks or languages, cannot generalize well, when sharing with a loosely related task. Such case, which we call **sparse transfer**, might actually hurt performance, a phenomenon known as negative transfer. Our contribution is using adversarial training across tasks, to "soft-code" shared and private spaces, to avoid the shared space gets too sparse. In CLL, our proposed architecture considers another challenge of dealing with low-quality input.

1 Introduction

Transfer learning in neural networks has been applied in recent years to improving the performance of related tasks, for example, 1) multi-task learning (MTL) with different tasks (labeled data available all tasks) and 2) cross-lingual learning (CLL) with different language (but the same task) though labeled data available only in source language. For both settings, one of their most common strategies is hard-parameter sharing, as shown in Figure 1a, which shares the hidden layers across tasks, which we will call **shared layer**. This approach works well when tasks are closely related, when most features are invariant across tasks. Otherwise, which we call **sparse transfer**, transferring between loosely related tasks often hurt performance, known as negative transfer. We elaborate this problem in MTL and CLL scenarios.

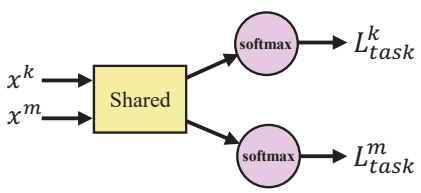
First, for MTL, the shared space is reported to be sparse, in an architecture with one shared encoder (Sachan and Neubig, 2018), when shared by K (*e.g.*, $K > 2$ tasks) loosely-related tasks. To address this problem, as shown in Figure 1b, recent models (Liu et al., 2017; Lin et al., 2018) divide the features of different tasks into task-invariant and task-dependent latent spaces, which we will call **shared** and **private** spaces from this point on. However, since such approach still hard-codes shared and private features, deciding which subsets of tasks should share encoders in many-task settings, among all possible combinations of tasks, is a non-trivial design problem (Ruder et al., 2019; Sanh et al., 2019).

Second, for CLL, the given task in source language (with rich resources) transfers to that for target languages without training resources. For the latter, machine-translated resources are fed instead, to the shared encoder (Schwenk and Douze, 2017; Conneau et al., 2018). When translation is perfect, the shared space would be dense: For example, English training pair with *entailment* relationship, "Because it looked so formidable" and "It really did look wonderful" can be translated to Chinese sentences of the same meaning, to preserve labels. Meanwhile, its translation into "因为它看起来那么可怕" (Because it looks so scary) and "它真的看起来很棒" (It really looks great), fails to preserve the entailment relationship, and makes the shared space sparse.

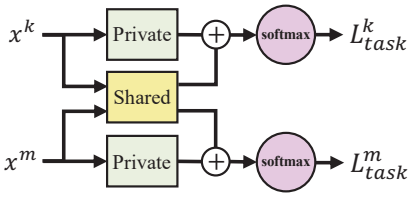
As a unified solution for both problems, we propose **soft-coding** approaches that can adapt in the following novel ways.

First, for MTL, we propose Task-Adaptive Representation learning using Soft-coding, namely TARS, wherein shared and private features are both *mixtures of features*. Specifically, as shown in Figure 1c, TARS begins as a generic sharing framework using one common shared encoder, but also adopts its paired task-specific layers to feed a Mixture-of-Experts (MoE) module (Shazeer et al., 2017; Guo et al., 2018) which captures soft-private features with a weighted combination of all task-dependent features, where

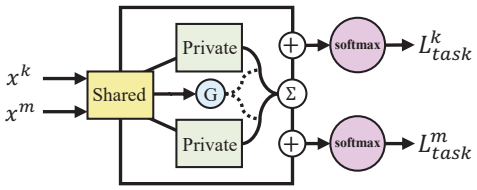
Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1560–1568
Florence, Italy, July 28 - August 2, 2019.



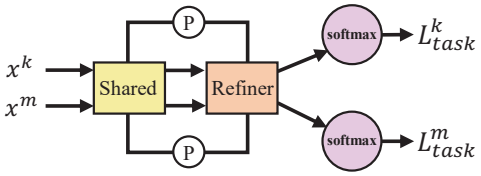
(a) Fully-Shared Model (FS)



(b) Shared-Private Model (SP)



(c) TARS



(d) CASE

Figure 1: Illustration of transfer learning architectures. Yellow and green boxes represent shared and private LSTM layers. G and P indicates a gating network and a policy network respectively.

a gating network G in Figure 1c, decides on output weights for each task. Based on this basic architecture, TARS softly-shares features balanced by two *conflicting auxiliary losses*: one is used to eliminate private features from the shared space, which decreases the generalization across task, while the other is used to keep shared space "dense" with soft-private features, which is a form of adversarial training. Such balancing efforts prevent the shared space from being too sparse to be generalized for every task, even when $K > 2$.

Second, for CLL, we propose a Cross-lingual AdverSarial Example, namely CASE. Compared to Figure 1c, task-specific private layers no longer exist in Figure 1d, because CLL deals with a single task for multiple languages. Instead, for an additional challenge of refining low-quality input, we add *Refiner*. Specifically, once the source language is translated into the target language, CASE moves the noisy representation on the target side towards a direction of space on the source side back in a form of adversarial example, and uses this as an additional training data to task classifier. However, this refinement may have adverse effects (Yeo et al., 2018), for which a policy network P in Figure 1d decides whether to refine or not.

To demonstrate the effectiveness and flexibility of our soft-coding approaches, we evaluate TARS on five different datasets covering diverse scenarios and CASE on cross-lingual natural language inference (XNLI) datasets with 15 languages (including low-resource language such as Swahili and Urdu), and show that TARS and CASE outperform existing hard-coding approaches.

2 Preliminaries

2.1 Problem Statement

Formally, we assume the existence of K datasets $\{\mathcal{D}^k\}_{k=1}^K$, where each $\mathcal{D}^k$ contains $|\mathcal{D}^k|$ data samples for classification task k. Specifically,

$$\mathcal{D}^k = \{(x_i^k, y_i^k)\}_{i=1}^{|\mathcal{D}^k|} \quad (1)$$

where x_i^k and y_i^k denote a sentence (or pair) and its corresponding label for task k. In CLL, $\mathcal{D}^k$ is given only for one language, for which we create a new dataset $\tilde{\mathcal{D}}^k = \{(\tilde{x}_i^k, y_i^k)\}$, where $\tilde{x}_i^k$ is translated, using neural machine translation (NMT), for training task k (for another language). Transfer learning aims to improve classification by learning these K tasks in parallel. Thus, our objective is to learn a sentence (or pair) representation $\mathbf{x}^k$ per task k, but take into account the correlation among related tasks.

Specifically, given an input sequence $x^k = \{w_1^k, w_2^k, ..., w_T^k\}$ with length T, we aim to learn a sentence representation $\mathbf{x}^k$ for the entire sequence as follows, $\mathbf{x}^k = \texttt{Encoder}(\{w_1^k, w_2^k, ..., w_T^k\})$. Following (Conneau et al., 2017), the final output representation $\mathbf{x}^k$ is ultimately fed into a corresponding classifier which consists of multiple fully connected layers culminating in a softmax layer, *i.e.*, $\hat{y}^k = \texttt{softmax}(\mathbf{W}^k\mathbf{x}^k + \mathbf{b}^k)$. The parameters of the network are trained to minimize the loss L_{task} of the predicted and true distribu-

tions on all the tasks as follows:

$$L_{task} = \sum_{k=1}^{K} L(\hat{y}^k, y^k) \tag{2}$$

where $L(\hat{y}^k, y^k)$ denote a typical cross-entropy loss for each task k.

2.2 Baseline: Hard-code Approach

As overviewed in Section 1, the success of transfer learning depends on the sharing scheme in latent feature space. Existing architectures differ in how to group the shared features to maximize sharing, as illustrated in Figure 1. We overview the existing approaches into the following two categories.

Base I: Fully-Shared Model (FS) As shown in Figure 1a, the Fully-Shared (FS) model adopts a single shared encoder `S-Encoder` to extract features generalized for all the tasks. For example, given two tasks k and m, all features $\mathbf{s}^k$ of task k are expected to be shared by task m and vice versa, *i.e.*, $\mathbf{s}^k = \texttt{S-Encoder}(\{w_1^k, w_2^k, ..., w_T^k\}; \theta_s)$, where θ_s represents the parameters of the shared encoder. In FS model, $\mathbf{s}^k$ is equivalent to $\mathbf{x}^k$ fed into classifiers.

Base II: Shared-Private Model (SP) As Figure 1b shows, the Shared-Private (SP) model consists of two modules: (1) the underlying shared encoder `S-Encoder` responsible to capture task-invariant features, and (2) the private encoder `P-Encoder` to extract task-dependent features, *i.e.*, $\mathbf{p}^k = \texttt{P-Encoder}(\{w_1^k, w_2^k, ..., w_T^k\}; \theta_p^k)$, where θ_p^k represents the parameters of each private encoder. Then, both shared representation $\mathbf{s}^k$ and private representation $\mathbf{p}^k$ are concatenated to construct the final sentence representation: $\mathbf{x}^k = \mathbf{s}^k \oplus \mathbf{p}^k$.

These hard-code approaches greatly reduce the risk of overfitting to capture all of the tasks simultaneously, but have the caveat that the ability of shared space to model task-invariant features can be significantly reduced (Sachan and Neubig, 2018). We empirically show our observations are consistent in Section 5.2.

3 Soft-code Approach for MTL: TARS

Inspired by the limitation of hard-coding approaches, our proposed model, TARS, begins with FS model but progressively adapts to task characteristics, as shown in Figure 1c.

Soft-Private Module TARS first models the multiple tasks as MoE, where each task has an individual expert network, and weighs the experts for different task examples. To be specific, TARS feeds the shared features $\mathbf{s}^k$ into individual `P-Encoder` for each task, to encode task-dependent features as follows:

$$\mathbf{p}^k = \texttt{P-Encoder}(\mathbf{s}^k; \theta_p^k) \tag{3}$$

Simultaneously, a gating network decides on output weights for each expert (*i.e.*, individual `P-Encoder`). Specifically, the gating network G, parameterized by θ_g, is used to map the shared representation of current task into the correct expert, and each expert is thus learning task-dependent features for that task, estimating task label of $\mathbf{s}^k$:

$$G(\mathbf{s}^k; \theta_g) = \texttt{softmax}(\mathbf{W}_g \mathbf{s}^k + \mathbf{b}_g) \tag{4}$$

where $\mathbf{W}_g$ and $\mathbf{b}_g$ is a trainable weight matrix and a bias, respectively. Based on above, the final soft-private representation $\mathbf{p}(\mathbf{s}^k)$ is a mixture of all expert outputs with respect to $\mathbf{s}^k$ as the following:

$$\mathbf{p}(\mathbf{s}^k) = \sum_{k=1}^{K} G(\mathbf{s}^k; \theta_g) \cdot \mathbf{p}^k \tag{5}$$

Soft-Shared Module In order to learn task-invariant features, inspired by (Liu et al., 2017), TARS adopts an adversarial network, which contains a feature extractor and a task discriminator D. The basic idea is to learn features that cannot be distinguished by D. Specifically, D aims to discriminate which task the feature comes from, while the feature extractor (*e.g.*, `S-Encoder`) tries to fool D so that it cannot identify the task of the feature and is hence task-invariant. More formally,

$$L_{adv} = \min_{\theta_s} \lambda \max_{\theta_d} \sum_{k=1}^{K} \sum_{i=1}^{|\mathcal{D}^k|} d_i^k log[D(\mathbf{s}^k; \theta_d)] \tag{6}$$

where d_i^k is the ground-truth task label, θ_d is the parameter of task discriminator D, and λ is a hyperparameter. As mentioned before, such adversarial learning has been verified to be very effective for extracting task-invariant features. However, trying to keep the shared space too pure inevitably leads to sparseness, for which we additionally introduce the density constraint L_{dense} for this purpose.

Specifically, the objective of the density constraint L_{dense} is to push the soft-private features from the private embeddings closer to the shared ones, such that the shared space is encouraged to being dense rather than being too sparse, resolving the sparseness of the shared space. Therefore, the soft-shared features might be more informative in this case. Formally,

$$L_{dense} = \sum_{k=1}^{K} ||\mathbf{p}(\mathbf{s}^k) - \mathbf{s}^k||^2 \quad (7)$$

where $||\cdot||^2$ is the mean squared L-2 norm.

Training and Inference Lastly, the soft-private and soft-shared representations $\mathbf{p}(\mathbf{s}^k)$ and $\mathbf{s}^k$ are concatenated, *i.e.*, $\mathbf{x}^k = \mathbf{s}^k \oplus \mathbf{p}(\mathbf{s}^k)$, to feed the all networks in TARS with the following loss:

$$L_{TARS} = L_{task} + L_{adv} + L_{dense} \quad (8)$$

TARS is trained with backpropagation, and adopts a gradient reversal layer (Ganin and Lempitsky, 2015) to address minimax optimization problem. Note that, unlike hard-code approaches, zero-shot learning is also possible since TARS can adapt to a new target task (*e.g.*, cross-domain or -lingual), by aligning it with the trained expert gate deciding what combination of the expert to use in Eq. (4) and Eq. (5) on inference.

4 Soft-code Approach for CLL: CASE

This section revises L_{dense} in Eq. (7) for CLL scenario. Note that, in CLL, sparse space corresponds to mistranslated low-resource language, which we call **pseudo-sentence**. The goal of L_{dense} is thus replaced by, softly correcting the representation to align better L_{align}, while preserving the semantics L_{sem}. For that purpose, we propose a *Refiner* replacing L_{dense} with these two new losses.

Refinement by Perturbation We first discuss how to refine pseudo-sentences by perturbation Δ for higher learning effectiveness. Related ideas are ensuring the robustness of a model, by finding Δ that changes a prediction, or, $f(x) = y$ while $f(x+\Delta) \neq y$ (Goodfellow et al., 2015). Inspired, CASE explores if incorrect translations that may cause wrong predictions in the target language can be moved back to change predictions.

For which, based on the basic architecture of variational auto-encoder (VAE) (Kingma and Welling, 2013), CASE models a neural refiner to refine low-quality representations. Specifically, as shown in Figure 1d, CASE first encodes pseudo-parallel sentences into shared space, *e.g.*, $(\mathbf{x}, \tilde{\mathbf{x}})$. Then, the refiner which consists of two encoding feed-forward network $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ converts the representations into two distribution variables $\mu(\tilde{\mathbf{x}})$ and $\sigma(\tilde{\mathbf{x}})$, the mean and standard deviation for pseudo representations. Unlike traditional VAE minimizing the latent loss that measures how closely the latent variables match a unit Gaussian, *i.e.*, $\mathbb{KL}(\mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x})), \mathcal{N}(0, 1))$, CASE enhances the latent loss with the pseudo-parallel representation, to generate pseudo-adversarial example $\tilde{\mathbf{z}}$ that roughly follows a representation $\mathbf{x}$ from resource-rich space as follows:

$$L_{align} = \mathbb{KL}(\mathcal{N}(\mu(\tilde{\mathbf{x}}), \sigma(\tilde{\mathbf{x}})), \mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x}))) \quad (9)$$

In order to optimize the KL divergence, CASE applies a simple reparameterization trick (Kingma and Welling, 2013). Using this trick, pseudo-adversarial example $\tilde{\mathbf{z}}$ is generated from the mean and standard deviation vectors, *i.e.*, $\tilde{\mathbf{z}} = \mu(\tilde{\mathbf{x}}) + \sigma(\tilde{\mathbf{x}}) \cdot \epsilon$, where $\epsilon \in \mathcal{N}(0, 1)$. This constraint not only allows us to generate an informative representation, but also improves the generalization of our network, towards $\mathbf{x}$ (*e.g.*, English) with higher confidence. Then, CASE aims at preserving its original semantics in the latent space, for which CASE includes the reconstruction loss, which is a mean squared error, to measure how accurately the pseudo-adversarial example $\tilde{\mathbf{z}}$ preserves its original semantics. *i.e.*, $L_{sim} = \Sigma_{|\tilde{\mathcal{D}}|}||\tilde{\mathbf{z}} - \tilde{\mathbf{x}}||^2$. As a result, $\tilde{\mathbf{z}}$ is fed into the classifier, and the overall loss of CASE is defined as follows:

$$\begin{aligned} L_{CASE} = L_{task} + L_{adv} \\ + L_{align} + L_{sim} \end{aligned} \quad (10)$$

Selective Refinement Lastly, CASE aims to refine only when the perturbation can refine the translation. In other words, if the translation is already good, CASE avoids a refinement, by parameterizing refinement with α set to be near zero. Not applying a refinement for correct translation is important, since more than half of translations is correctly translated, as reported by (Yeo et al., 2018), such that refinement may lower the quality.

For computing α, CASE adapts a policy network P, which consists of a feed forward network $\mathcal{P}(\mathbf{x}; \theta_p) = \texttt{softmax}(\mathbf{W}_p\mathbf{x} + \mathbf{b}_p)$, to identify wrong translations by capturing the difference of domain distribution. Then, the policy is calculated

as follows:

$$\alpha = \mathbb{KL}(\mathcal{P}(\tilde{\mathbf{x}})||\mathcal{P}(\mathbf{x})) = \sum_{\mathbf{x}\in\mathcal{D},\tilde{\mathbf{x}}\in\tilde{\mathcal{D}}} \mathcal{P}(\tilde{\mathbf{x}}) \log \frac{\mathcal{P}(\tilde{\mathbf{x}})}{\mathcal{P}(\mathbf{x})} \tag{11}$$

in which $\mathcal{P}(\mathbf{x})$ outputs a domain distribution of $\mathbf{x}$, and CASE estimates α as the difference between two distributions (*i.e.*, KL divergence), and the final loss function is defined factoring in α: $L_{CASE} = L_{task} + L_{adv} + \alpha(L_{align} + L_{sim})$.

5 Experiments

5.1 Experimental Settings

To show the effectiveness of our proposed approaches, we conduct experiments on both multi-task and cross-lingual settings.

Multi-task Dataset For Multi-task learning, we use five different datasets on Natural Language Inference (NLI) and Paraphrase Identification (PI) tasks: SNLI (Bowman et al., 2015), MNLI (Williams et al., 2018), and CNLI[1], for single-domain-English, multi-domain-English, and Chinese NLI respectively; QQP (Csernai et al., 2017) and LCQMC (Liu et al., 2018) for English and Chinese PI.

Cross-lingual Dataset We use the cross-lingual natural language inference (XNLI) dataset (Conneau et al., 2018)[2] from 15 different languages for Cross-lingual learning. The dataset is a version of MNLI (Williams et al., 2018) where 2,500 dev and 5,000 test sets have been translated (by humans) into 14 languages. For training datasets, the English training data is translated into each target language by NMT.

Implementation Details For all encoder, we adopt BiLSTM-max (Conneau et al., 2017) model and the pre-trained word embeddings we use are 300-dimensional fastText word embeddings (Bojanowski et al., 2017). Following (Conneau et al., 2018), the BiLSTM hidden states is set to 256 and Adam optimizer with a learning rate of 0.001 was applied. The learning rate was decreased by a factor of 0.85 when the target dev accuracy does not improve. As in (Conneau et al., 2018), for text classification networks, we use a feed-forward neural network with one hidden layer of 128 hidden units with a dropout rate of 0.1, to measure the relatedness of a given premise and hypothesis. The hyperparameter λ is empirically set to 0.005. All our implementation is available at `github.com/haejupark/soft`.

[1] https://github.com/blcunlp/CNLI

[2] https://www.nyu.edu/projects/bowman/xnli/XNLI-1.0.zip

Source	Model	SNLI	MNLI	QQP
(Single Task)	BiLSTM-max	81.95	65.98	85.89
SNLI+MNLI	AFS	82.06 (+0.11)	66.51 (+0.53)	-
	ASP	82.28 (+0.33)	67.39 (+1.41)	-
	TARS	**82.67** (+0.70)	**67.79** (+1.81)	-
SNLI+QQP	AFS	82.03 (+0.08)	-	85.08 (-0.81)
	ASP	82.20 (+0.25)	-	86.22 (+0.33)
	TARS	**82.54** (+0.59)	-	**86.51** (+0.62)
MNLI+QQP	AFS	-	66.62 (+0.64)	85.59 (-0.30)
	ASP	-	66.92 (+0.94)	86.12 (+0.23)
	TARS	-	**67.37** (+1.39)	**86.47** (+0.58)

Table 1: Accuracy over MTL with two-source tasks

5.2 Experimental Result I: MTL

Using (Liu et al., 2017) as hard-code baselines, we apply Adversarial training (and so-called orthogonality constraints) to FS and SP models, namely AFS and ASP. Such techniques enhance the distinct nature of shared and private features.

Two-source MTL Table 1 shows the performance on three text classification tasks. The first row shows the results of "single task", and other rows show the results of "multiple tasks" by corresponding MTL models trained with two source tasks. More concretely, (SNLI+MNLI) and (*NLI+QQP) are for cross-domain and cross-task classification respectively. In this table, we can see that TARS achieves higher accuracy than all sharing scheme baselines in all scenarios, surpassing multi-task learning (*i.e.*, ASP) as well as single task learning. These results show that our soft-code approach also works well in typical MTL settings with two source tasks, though they are not our targeted sparse scenario.

Three-source MTL In Table 2, MTL models use three source tasks (SNLI+MNLI+QQP), where the first row shows the results of "single task". We first test SNLI, MNLI, and QQP as a supervised target task. From the results, we can see that TARS outperforms all baselines including MoE, which is a variant of TARS excluding the two auxiliary losses. We also include the recent work,

Model	SNLI	MNLI	QQP	CNLI	LCQMC
BiLSTM-max	81.95	65.98	85.89	64.42	79.69
AFS	81.70 (-0.25)	66.78 (+0.80)	85.41 (-0.48)	39.70 (-24.72)	61.29 (-18.40)
ASP	82.23 (+0.28)	66.92 (+0.94)	86.04 (+0.15)	-	-
MoE	81.55 (-0.40)	66.72 (+0.74)	85.23 (-0.66)	39.45 (-24.97)	63.02 (-16.67)
MMoE	81.46 (-0.49)	67.01 (+1.03)	85.29 (-0.60)	-	-
TARS	**83.12** (+1.17)	**68.24** (+2.26)	**86.15** (+0.26)	**40.52** (-23.90)	**63.45** (-16.24)

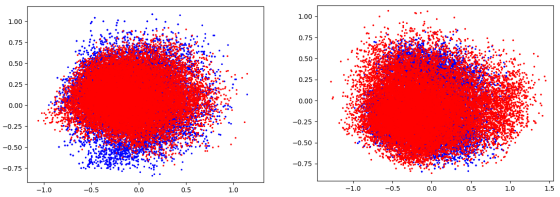
Table 2: Accuracy of MTL with three-source tasks

MMoE (Ma et al., 2018), which explicitly learns to model task relationship by modeling an expert for each task (which is not desirable for a new task). This suggests that the synergetic effect of soft-private and -shared modules in TARS is critical to outperform other baselines.

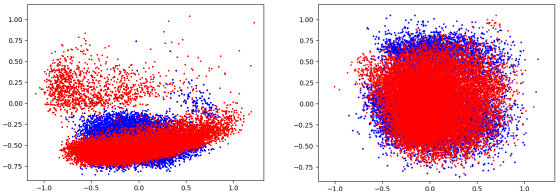
Specifically, AFS and ASP show a "negative transfer", which is an inherent challenge of MTL. For example, ASP with three-source tasks achieves 82.23% and 66.92% accuracy, respectively, in SNLI and MNLI, which are lower than 82.28% and 67.39% accuracy with its best performance with two-source tasks. In contrast, TARS overcomes such challenges, for example, $83.12\% > 82.67\%$ and $68.24\% > 67.79\%$ in SNLI and MNLI, except for QQP, which can be further improved by asymmetric MTL techniques (Lee et al., 2016).

To investigate how TARS helps transfer knowledge across tasks, Figure 2a and 2b contrast the feature representation of shared space in ASP and TARS, in two- and three-source settings respectively. First, for two-sources, ASP and TARS are comparable, capturing the distribution of two tasks that are nearly identical, which is desirable for transfer learning. Second, for three sources, the shared space of ASP shows two quite distinct distributions (task-dependent), while TARS keeps two distributions comparable (and task-invariant).

Zero-shot Learning Lastly, in Table 2, we test zero-shot learning with two target tasks, CNLI and LCQMC, excluding their own training data (except for the first row single task). As ASP requires target task labels to train its private encoders, we compare TARS only with AFS and MoE, where TARS shows the best performance in MTL. As shown in Figure 3, we observe that when TARS covers sentences in CNLI and LCQMC, using its gating network that identifies that the unknown target tasks are the most similar to SNLI and QQP, respectively: Specifically, highest weights are assigned to these two, but other source tasks also contribute, with non-zero weights.



(a) Shared space for two-source



(b) Shared space for three-source

Figure 2: PCA visualization. Blue and red indicate the shared features of SNLI and QQP, respectively, using ASP (left) and TARS (right).

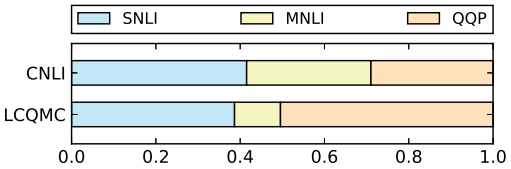


Figure 3: Gating weights in zero-shot learning.

5.3 Experimental Result II: CLL

Table 3 shows our results on 14 XNLI languages. Following (Conneau et al., 2018), we divide the models into following three categories: 1) *Translate train*, where the English NLI training set is translated into each XNLI language and train a language-specific NLI classifier for each language; 2) *Translate test*, where all dev and test set of XNLI is translated to English and apply English NLI classifier; and 3) *Zero-shot Learning*, where English classifier is directly applied to the target language without any translation. We also report the results of XNLI baselines (Conneau et al., 2018), a supervised cross-lingual MTL model that combines the L_{adv} loss using pseudo-parallel data (Liu et al., 2017), the multilingual BERT (Devlin et al., 2018), and the recent work of (Artetxe and Schwenk, 2018).

First, in Table 3, we can see that BiLSTM model (Conneau et al., 2018), in *Translate test*, appears

	en → xx													
	fr	es	de	el	bg	ru	tr	ar	vi	th	zh	hi	sw	ur
Translate train, each NLI models for each language														
BiLSTM (Conneau et al., 2018)	68.3	68.8	66.5	66.4	67.4	66.5	64.5	65.8	66.0	62.8	67.0	62.1	58.2	56.6
BiLSTM+MTL (Liu et al., 2017)	66.0	68.7	67.3	67.4	68.2	64.8	65.3	65.1	66.1	59.3	66.2	54.2	60.0	58.0
CASE (w/o selective)	70.4	70.3	70.2	69.2	70.0	69.6	69.4	68.8	69.3	67.4	70.9	67.4	67.9	66.8
CASE (w selective)	71.1	71.2	70.0	70.3	69.9	69.8	70.0	70.1	70.5	68.9	71.3	68.7	67.7	67.5
Multilingual BERT (Devlin et al., 2018)	-	77.3*	75.2*	-	-	-	-	70.5*	-	-	74.2*	-	-	61.7*
Multilingual BERT on CASE (w selective)	**78.7**	**78.2**	**76.4**	**76.7**	**75.8**	**75.5**	**73.3**	**73.7**	**74.2**	**72.3**	**74.3**	**72.2**	**71.6**	**71.3**
Translate test, one English NLI model for all languages														
BiLSTM (Conneau et al., 2018)	70.4	70.7	68.7	69.1	70.4	67.8	66.3	66.8	66.5	64.4	68.3	64.2	61.8	59.3
Multilingual BERT (Devlin et al., 2018)	-	74.9*	74.4*	-	-	-	-	70.4*	-	-	70.1*	-	-	62.1*
Zero-Shot Learning, one NLI model for all languages														
BiLSTM (Conneau et al., 2018)	67.7	68.7	67.7	68.9	67.9	65.4	64.2	64.8	66.4	64.1	65.8	64.1	55.7	58.4
Multilingual BERT (Devlin et al., 2018)	-	74.3*	70.5*	-	-	-	-	62.1*	-	-	63.8*	-	-	58.3*
(Artetxe and Schwenk, 2018)	71.9	72.9	72.6	73.1	74.2	71.5	69.7	71.4	72.0	69.2	71.4	65.5	62.2	61.0

Table 3: Accuracy over 14 XNLI languages (test set accuracy). We report results for translation baselines, multi-task learning baselines and zero-shot baselines. Overall best results are in bold, and the best in each group is underlined. All results * from its Github project https://github.com/google-research/bert/blob/master/multilingual.md.

to perform consistently better than *Translate train* for all languages, which means a single English model works better than training each target model with translated data. In contrast, Multilingual BERT (Devlin et al., 2018) achieves best results on *Translate train*, outperforming most languages, suggesting the generalization of BERT across languages significantly better than BiLSTM model.

Meanwhile, CASE, significantly outperforms the BiLSTM and BiLSTM+MTL models in *Translate train* for all languages, and even outperforms BiLSTM in *Translate test*. Compared to the best performing MTL baseline, CASE achieves an improvement of 1.7% and 9.5% in Bulgarian (bg) and Urdu (ur) languages respectively. From these results, we observe that: 1) the improvements on low-resource language (*e.g.*, Swahili and Urdu) are more substantial than those on other languages; 2) the selective refinement strategy consistently contributes to the performance improvement. These results show that CASE, by incorporating pseudo-adversarial example as an additional resource, contributes to the robustness and the generalization of the model.

Lastly, we show that CASE with multilingual BERT model achieves the state-of-the-art, and even significantly outperforms the supervised approach of (Artetxe and Schwenk, 2018) enjoying an unfair advantage of extremely large amounts of parallel sentences. These results show that CASE, with the help of strong baselines, gets a significant boost in performance, particularly for Swahili and Urdu that are low-resource languages, achieving the improvement of 9.4% and 10.3% respectively.

Robustness Analysis In order to verify whether CASE is robust, inspired by (Goodfellow et al., 2015), we test if models keep its prediction, even after changes to the sentence, as long as the meaning remains unchanged. For example, the given sentence can be paraphrased by changing some words with their synonyms, and the models should give the same answer to the paraphrase.

Meanwhile, existing models, especially those overfitted to surface forms, are sensitive to such "semantic-preserving" perturbations. As human annotation for such perturbations is expensive, an automated approach (Alzantot et al., 2018) was studied for English, to generate semantic-preserving adversaries that fool well-trained sentiment analysis and NLI models with success rates of 97% and 70%, respectively. In our problem setting of XNLI, we need such a generator (or generated resources) for each language. For which, we identify three research questions:

- **(RQ1)** How hard is it to build a generator for a new language?
- **(RQ2)** Are the observations consistent?
- **(RQ3)** Does our model improve robustness?

Specifically, in this paper we focus on Chinese, as we could hire native speaking volunteers to validate whether automatically generated perturbations indeed preserve semantics.

First, for RQ1, we leverage Chinese synonyms and antonyms to build counter fitting vectors as (Mrkšić et al., 2016) to ensure the selected words are synonyms. Then, we slightly change

Original Text Prediction: **Contradiction** (Confidence = 97%)
Premise: 能帮助我的女孩在小镇的另一边。 Hypothesis: **没有**人能帮助我。

Adversarial Text Prediction: **Entailment** (Confidence = 59%)
Premise: 能帮助我的女孩在小镇的另一边。 Hypothesis: **并没有**人能帮助我。

Table 4: Example of generated adversarial example for chinese natural language inference task.

(Alzantot et al., 2018)[3] to automatically generate Chinese perturbations for NLI task. Following the convention of (Alzantot et al., 2018), for NLI problem, we only add perturbation to the hypothesis, excluding premise, and aim to divert the prediction result from **entailment** to **contradiction**, and vice versa. Table 4 is an example of generated adversarial example.

For RQ2, we validate the automatically generated perturbations by native speaking volunteers. We show volunteers 500 samples to label whether it is contradiction, neutral or entailment. 84 percent of the responses matched the original ground truth. Second, we sample 500 samples, with each sample including the original sentence and the corresponding adversarial example. Volunteers were asked to judge the similarity of each pair on a scale from 1 (very different) to 4 (very similar). The average rating is 2.12, which shows the performance of our implementation for Chinese perturbation is also competitive.

Lastly, for RQ3, we show the attack success rates over generated adversarial example in Table 5. For comparison, we include the single task and MTL baselines. As shown in the Table 5, CASEs are able to achieve higher defense rate (or lower success rate) in performance of 36.6%, while baselines obtained 15.7% and 21.4% respectively, which demonstrates incorporating pseudo-adversarial example is indeed helpful to the robustness of the model.

Model	% Success
BiLSTM	0.843
BiLSTM+MTL	0.786
CASE (w/o selective)	0.657
CASE (w selective)	0.634

Table 5: Attack success rates over Chinese adversarial example for the text classification task.

[3]https://github.com/nesl/nlp_adversarial_examples

6 Related Work

Transfer Learning: Transfer learning enables effective knowledge transfer from the source to the target task. Early works mainly focused on the shared representation methods (Liu et al., 2017; Tong et al., 2018; Lin et al., 2018), using a single shared encoder between all tasks while keeping several task-dependent output layers. However, the sparseness of the shared space, when shared by K tasks, was observed (Sachan and Neubig, 2018). In this paper, we study a soft-coding approach to overcome sparsity, leading to performance gains in MTL and CLL tasks. Closely related work is MMoE (Ma et al., 2018), which explicitly learns the task relationship by modeling a gating network for each task. Such work does not consider which combination of networks to use for a new task, while we differentiate by deciding such combination for a new task based on its similarity to the source tasks.

Adversarial Example: Despite the success of deep neural networks, neural models are still brittle to adversarial examples (Goodfellow et al., 2015). Recently, adversarial examples are widely incorporated into training to improve the generalization and robustness of the model using back-translated paraphrases (Iyyer et al., 2018), machine-generated rules (Ribeiro et al., 2018), black-box (Alzantot et al., 2018) and white-box (Ebrahimi et al., 2018). Inspired, we study pseudo-adversarial example in latent space to improve the robustness of the model. To the best of our knowledge, we are the first proposing pseudo-adversarial training in latent space for transfer learning.

7 Conclusion

In this paper, we study the limitations of hard-parameter sharing in sparse transfer learning. We propose soft-code approaches to avoid the sparseness observed in MTL and CLL. We have demonstrated the effectiveness and flexibility of our soft-code approaches in extensive evaluations over MTL and CLL scenarios.

Acknowledgments

This work is supported by Microsoft Research Asia and IITP grant funded by the Korean government (MSIT, 2017-0-01779, XAI). Hwang is a corresponding author.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *EMNLP*.

Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. In *arXiv preprint arXiv:1812.10464*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *EMNLP*.

Kornél Csernai, Shankar Iyer, and Nikhil Dandekar. 2017. Quora question pairs.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *ACL*.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *ICML*.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *ICLR*.

Jiang Guo, Darsh Shah, and Regina Barzilay. 2018. Multi-source domain adaptation with mixture of experts. In *EMNLP*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *NAACL-HLT*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Giwoong Lee, Eunho Yang, and Sung Hwang. 2016. Asymmetric multi-task learning based on task relatedness and loss. In *ICML*.

Ying Lin, Shengqi Yang, Veselin Stoyanov, and Heng Ji. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. In *ACL*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *ACL*.

Xin Liu, Qingcai Chen, Chong Deng, Jing Chen, Dongfang Li, and Huajun Zeng. 2018. LCQMC:A Large-scale Chinese Question Matching Corpus. In *COLING*.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *NAACL-HLT*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *ACL*.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2019. Latent multi-task architecture learning. In *AAAI*.

Devendra Singh Sachan and Graham Neubig. 2018. Parameter sharing methods for multilingual self-attentional translation models. In *WMT*.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2019. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *AAAI*.

Holger Schwenk and Matthijs Douze. 2017. Learning joint multilingual sentence representations with neural machine translation. *arXiv preprint arXiv:1704.04154*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint*.

Xiaowei Tong, Zhenxin Fu, Mingyue Shang, Dongyan Zhao, and Rui Yan. 2018. One "ruler" for all languages: Multi-lingual dialogue evaluation with adversarial multi-task learning. In *IJCAI*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*.

Jinyoung Yeo, Geungyu Wang, Hyunsouk Cho, Seungtaek Choi, and Seung-won Hwang. 2018. Machine-translated knowledge transfer for commonsense causal reasoning. In *AAAI*.