Multi-Task Deep Neural Networks for Natural Language Understanding

Xiaodong Liu[*1], **Pengcheng He**[*2], **Weizhu Chen**[2], **Jianfeng Gao**[1]
[1] Microsoft Research [2] Microsoft Dynamics 365 AI
{xiaodl,penhe,wzchen,jfgao}@microsoft.com

Abstract

In this paper, we present a Multi-Task Deep Neural Network (MT-DNN) for learning representations across multiple natural language understanding (NLU) tasks. MT-DNN not only leverages large amounts of cross-task data, but also benefits from a regularization effect that leads to more general representations to help adapt to new tasks and domains. MT-DNN extends the model proposed in Liu et al. (2015) by incorporating a pre-trained bidirectional transformer language model, known as BERT (Devlin et al., 2018). MT-DNN obtains new state-of-the-art results on ten NLU tasks, including SNLI, SciTail, and eight out of nine GLUE tasks, pushing the GLUE benchmark to 82.7% (2.2% absolute improvement) [1]. We also demonstrate using the SNLI and SciTail datasets that the representations learned by MT-DNN allow domain adaptation with substantially fewer in-domain labels than the pre-trained BERT representations. The code and pre-trained models are publicly available at https://github.com/namisan/mt-dnn.

1 Introduction

Learning vector-space representations of text, e.g., words and sentences, is fundamental to many natural language understanding (NLU) tasks. Two popular approaches are *multi-task learning* and *language model pre-training*. In this paper we combine the strengths of both approaches by proposing a new Multi-Task Deep Neural Network (MT-DNN).

Multi-Task Learning (MTL) is inspired by human learning activities where people often apply the knowledge learned from previous tasks to help learn a new task (Caruana, 1997; Zhang and Yang, 2017). For example, it is easier for a person who knows how to ski to learn skating than the one who does not. Similarly, it is useful for multiple (related) tasks to be learned jointly so that the knowledge learned in one task can benefit other tasks. Recently, there is a growing interest in applying MTL to representation learning using deep neural networks (DNNs) (Collobert et al., 2011; Liu et al., 2015; Luong et al., 2015; Xu et al., 2018; Guo et al., 2018; Ruder12 et al., 2019) for two reasons. First, supervised learning of DNNs requires large amounts of task-specific labeled data, which is not always available. MTL provides an effective way of leveraging supervised data from many related tasks. Second, the use of multi-task learning profits from a regularization effect via alleviating overfitting to a specific task, thus making the learned representations universal across tasks.

In contrast to MTL, language model pre-training has shown to be effective for learning universal language representations by leveraging large amounts of unlabeled data. A recent survey is included in Gao et al. (2018). Some of the most prominent examples are ELMo (Peters et al., 2018), GPT (Radford et al., 2018) and BERT (Devlin et al., 2018). These are neural network language models trained on text data using unsupervised objectives. For example, BERT is based on a multi-layer bidirectional Transformer, and is trained on plain text for masked word prediction and next sentence prediction tasks. To apply a pre-trained model to specific NLU tasks, we often need to fine-tune, for each task, the model with additional task-specific layers using task-specific training data. For example, Devlin et al. (2018) shows that BERT can be fine-tuned this way to create state-of-the-art models for a range of NLU tasks, such as question answering and natural language inference.

We argue that MTL and language model pre-training are complementary technologies, and can be combined to improve the learning of text rep-

[*] Equal Contribution.
[1] As of February 25, 2019 on the latest GLUE test set.

resentations to boost the performance of various NLU tasks. To this end, we extend the MT-DNN model originally proposed in Liu et al. (2015) by incorporating BERT as its shared text encoding layers. As shown in Figure 1, the lower layers (i.e., text encoding layers) are shared across all tasks, while the top layers are task-specific, combining different types of NLU tasks such as single-sentence classification, pairwise text classification, text similarity, and relevance ranking. Similar to the BERT model, MT-DNN can be adapted to a specific task via fine-tuning. Unlike BERT, MT-DNN uses MTL, in addition to language model pre-training, for learning text representations.

MT-DNN obtains new state-of-the-art results on eight out of nine NLU tasks [2] used in the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018), pushing the GLUE benchmark score to 82.7%, amounting to 2.2% absolute improvement over BERT. We further extend the superiority of MT-DNN to the SNLI (Bowman et al., 2015a) and SciTail (Khot et al., 2018) tasks. The representations learned by MT-DNN allow domain adaptation with substantially fewer in-domain labels than the pre-trained BERT representations. For example, our adapted models achieve the accuracy of 91.6% on SNLI and 95.0% on SciTail, outperforming the previous state-of-the-art performance by 1.5% and 6.7%, respectively. Even with only 0.1% or 1.0% of the original training data, the performance of MT-DNN on both SNLI and SciTail datasets is better than many existing models. All of these clearly demonstrate MT-DNN's exceptional generalization capability via multi-task learning.

2 Tasks

The MT-DNN model combines four types of NLU tasks: single-sentence classification, pairwise text classification, text similarity scoring, and relevance ranking. For concreteness, we describe them using the NLU tasks defined in the GLUE benchmark as examples.

Single-Sentence Classification: Given a sentence[3], the model labels it using one of the predefined class labels. For example, the **CoLA** task is to predict whether an English sentence is grammatically plausible. The **SST-2** task is to determine whether the sentiment of a sentence extracted from movie reviews is positive or negative.

Text Similarity: This is a regression task. Given a pair of sentences, the model predicts a real-value score indicating the semantic similarity of the two sentences. **STS-B** is the only example of the task in GLUE.

Pairwise Text Classification: Given a pair of sentences, the model determines the relationship of the two sentences based on a set of pre-defined labels. For example, both **RTE** and **MNLI** are language inference tasks, where the goal is to predict whether a sentence is an *entailment*, *contradiction*, or *neutral* with respect to the other. **QQP** and **MRPC** are paraphrase datasets that consist of sentence pairs. The task is to predict whether the sentences in the pair are semantically equivalent.

Relevance Ranking: Given a query and a list of candidate answers, the model ranks all the candidates in the order of relevance to the query. **QNLI** is a version of Stanford Question Answering Dataset (Rajpurkar et al., 2016). The task involves assessing whether a sentence contains the correct answer to a given query. Although QNLI is defined as a binary classification task in GLUE, in this study we formulate it as a pairwise ranking task, where the model is expected to rank the candidate that contains the correct answer higher than the candidate that does not. We will show that this formulation leads to a significant improvement in accuracy over binary classification.

3 The Proposed MT-DNN Model

The architecture of the MT-DNN model is shown in Figure 1. The lower layers are shared across all tasks, while the top layers represent task-specific outputs. The input X, which is a word sequence (either a sentence or a pair of sentences packed together) is first represented as a sequence of embedding vectors, one for each word, in l_1. Then the transformer encoder captures the contextual information for each word via self-attention, and gen-

[2] The only GLUE task where MT-DNN does not create a new state of the art result is WNLI. But as noted in the GLUE webpage (https://gluebenchmark.com/faq), there are issues in the dataset, and none of the submitted systems has ever outperformed the majority voting baseline whose accuracy is 65.1.

[3] In this study, a sentence can be an arbitrary span of contiguous text or word sequence, rather than a linguistically plausible sentence.

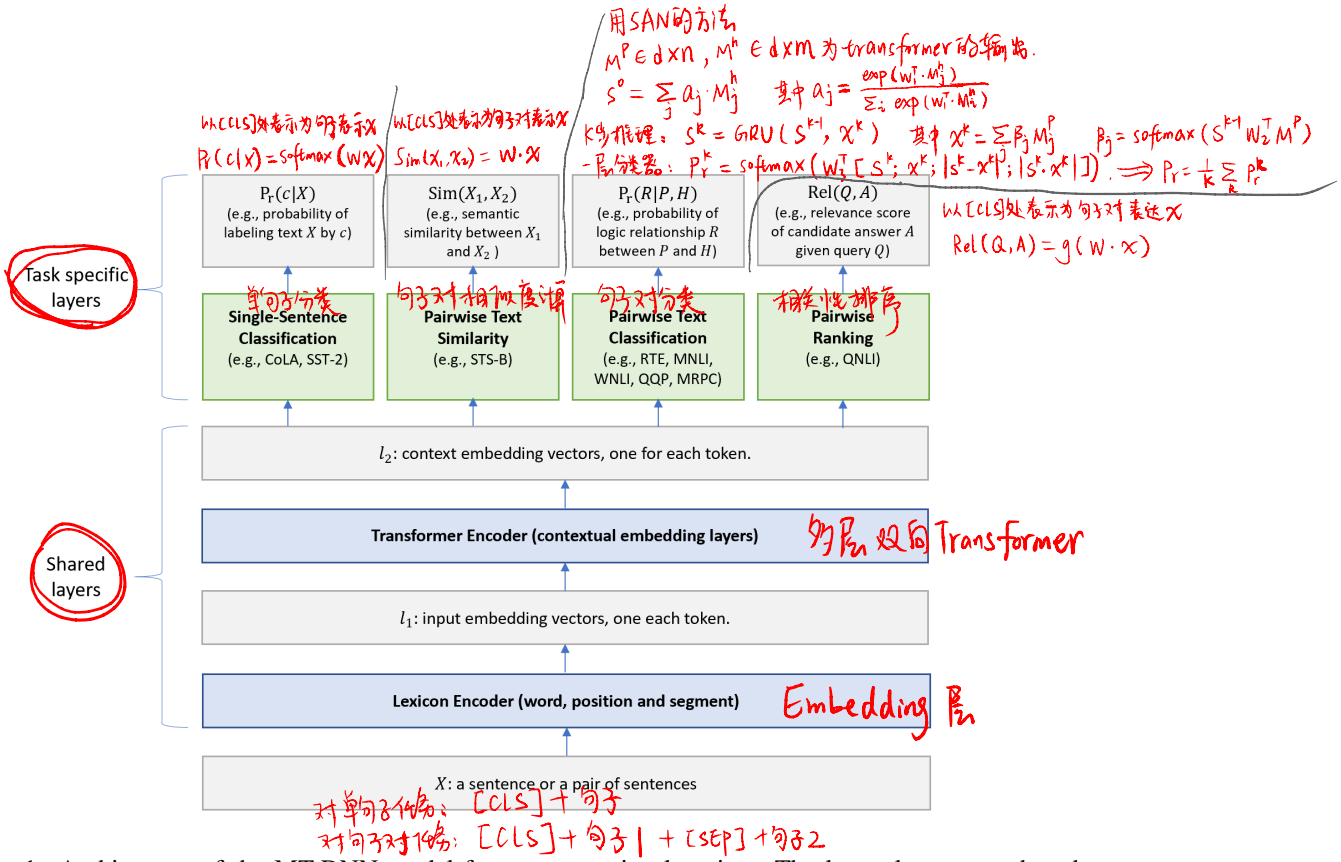

Figure 1: Architecture of the MT-DNN model for representation learning. The lower layers are shared across all tasks while the top layers are task-specific. The input X (either a sentence or a pair of sentences) is first represented as a sequence of embedding vectors, one for each word, in l_1. Then the Transformer encoder captures the contextual information for each word and generates the shared contextual embedding vectors in l_2. Finally, for each task, additional task-specific layers generate task-specific representations, followed by operations necessary for classification, similarity scoring, or relevance ranking.

erates a sequence of contextual embeddings in l_2. This is the shared semantic representation that is trained by our multi-task objectives. In what follows, we elaborate on the model in detail.

Lexicon Encoder (l_1): The input $X = \{x_1, ..., x_m\}$ is a sequence of tokens of length m. Following Devlin et al. (2018), the first token x_1 is always the `[CLS]` token. If X is packed by a sentence pair (X_1, X_2), we separate the two sentences with a special token `[SEP]`. The lexicon encoder maps X into a sequence of input embedding vectors, one for each token, constructed by summing the corresponding word, segment, and positional embeddings.

Transformer Encoder (l_2): We use a multi-layer bidirectional Transformer encoder (Vaswani et al., 2017) to map the input representation vectors (l_1) into a sequence of contextual embedding vectors $\mathbf{C} \in \mathbb{R}^{d \times m}$. This is the shared representation across different tasks. Unlike the BERT model (Devlin et al., 2018) that learns the representation via pre-training, MT-DNN learns the representation using multi-task objectives, in addition to pre-training.

Below, we will describe the task specific layers using the NLU tasks in GLUE as examples, although in practice we can incorporate arbitrary natural language tasks such as text generation where the output layers are implemented as a neural decoder.

Single-Sentence Classification Output: Suppose that $\mathbf{x}$ is the contextual embedding (l_2) of the token `[CLS]`, which can be viewed as the semantic representation of input sentence X. Take the SST-2 task as an example. The probability that X is labeled as class c (i.e., the sentiment) is predicted by a logistic regression with softmax:

$$P_r(c|X) = \mathrm{softmax}(\mathbf{W}_{SST}^{\top} \cdot \mathbf{x}), \quad (1)$$

where $\mathbf{W}_{SST}$ is the task-specific parameter matrix.

Text Similarity Output: Take the STS-B task as an example. Suppose that $\mathbf{x}$ is the contextual embedding (l_2) of `[CLS]` which can be viewed as the semantic representation of the input sentence pair (X_1, X_2). We introduce a task-specific parameter vector $\mathbf{w}_{STS}$ to compute the similarity score as:

$$\mathrm{Sim}(X_1, X_2) = \mathbf{w}_{STS}^{\top} \cdot \mathbf{x}, \quad (2)$$

where $\text{Sim}(X_1, X_2)$ is a real value of the range (-∞, ∞).

Pairwise Text Classification Output: Take natural language inference (NLI) as an example. The NLI task defined here involves a premise $P = (p_1, ..., p_m)$ of m words and a hypothesis $H = (h_1, ..., h_n)$ of n words, and aims to find a logical relationship R between P and H. The design of the output module follows the answer module of the stochastic answer network (SAN) (Liu et al., 2018a), a state-of-the-art neural NLI model. SAN's answer module uses multi-step reasoning. Rather than directly predicting the entailment given the input, it maintains a state and iteratively refines its predictions.

The SAN answer module works as follows. We first construct the working memory of premise P by concatenating the contextual embeddings of the words in P, which are the output of the transformer encoder, denoted as $\mathbf{M}^p \in \mathbb{R}^{d \times m}$, and similarly the working memory of hypothesis H, denoted as $\mathbf{M}^h \in \mathbb{R}^{d \times n}$. Then, we perform K-step reasoning on the memory to output the relation label, where K is a hyperparameter. At the beginning, the initial state $\mathbf{s}^0$ is the summary of $\mathbf{M}^h$: $\mathbf{s}^0 = \sum_j \alpha_j \mathbf{M}^h_j$, where $\alpha_j = \frac{\exp(\mathbf{w}_1^\top \cdot \mathbf{M}^h_j)}{\sum_i \exp(\mathbf{w}_1^\top \cdot \mathbf{M}^h_i)}$. At time step k in the range of $\{1, 2, , K-1\}$, the state is defined by $\mathbf{s}^k = \text{GRU}(\mathbf{s}^{k-1}, \mathbf{x}^k)$. Here, $\mathbf{x}^k$ is computed from the previous state $\mathbf{s}^{k-1}$ and memory $\mathbf{M}^p$: $\mathbf{x}^k = \sum_j \beta_j \mathbf{M}^p_j$ and $\beta_j = \text{softmax}(\mathbf{s}^{k-1} \mathbf{W}_2^\top \mathbf{M}^p)$. A one-layer classifier is used to determine the relation at each step k:

$$P^k_r = \text{softmax}(\mathbf{W}_3^\top [\mathbf{s}^k; \mathbf{x}^k; |\mathbf{s}^k - \mathbf{x}^k|; \mathbf{s}^k \cdot \mathbf{x}^k]). \quad (3)$$

At last, we utilize all of the K outputs by averaging the scores:

$$P_r = \text{avg}([P^0_r, P^1_r, ..., P^{K-1}_r]). \quad (4)$$

Each P_r is a probability distribution over all the relations $R \in \mathcal{R}$. During training, we apply *stochastic prediction dropout* (Liu et al., 2018b) before the above averaging operation. During decoding, we average all outputs to improve robustness.

Relevance Ranking Output: Take QNLI as an example. Suppose that $\mathbf{x}$ is the contextual embedding vector of `[CLS]` which is the semantic representation of a pair of question and its candidate answer (Q, A). We compute the relevance score as:

$$\text{Rel}(Q, A) = g(\mathbf{w}_{QNLI}^\top \cdot \mathbf{x}), \quad (5)$$

For a given Q, we rank all of its candidate answers based on their relevance scores computed using Equation 5.

3.1 The Training Procedure

The training procedure of MT-DNN consists of two stages: pretraining and multi-task learning. The pretraining stage follows that of the BERT model (Devlin et al., 2018). The parameters of the lexicon encoder and Transformer encoder are learned using two unsupervised prediction tasks: masked language modeling and next sentence prediction.[4]

In the multi-task learning stage, we use mini-batch based stochastic gradient descent (SGD) to learn the parameters of our model (i.e., the parameters of all shared layers and task-specific layers) as shown in Algorithm 1. In each epoch, a mini-batch b_t is selected(e.g., among all 9 GLUE tasks), and the model is updated according to the task-specific objective for the task t. This approximately optimizes the sum of all multi-task objectives.

For the classification tasks (i.e., single-sentence or pairwise text classification), we use the cross-entropy loss as the objective:

$$-\sum_c \mathbb{1}(X, c) \log(P_r(c|X)), \quad (6)$$

where $\mathbb{1}(X, c)$ is the binary indicator (0 or 1) if class label c is the correct classification for X, and $P_r(.)$ is defined by e.g., Equation 1 or 4.

For the text similarity tasks, such as STS-B, where each sentence pair is annotated with a real-valued score y, we use the mean squared error as the objective:

$$(y - \text{Sim}(X_1, X_2))^2, \quad (7)$$

where $\text{Sim}(.)$ is defined by Equation 2.

The objective for the relevance ranking tasks follows the pairwise learning-to-rank paradigm (Burges et al., 2005; Huang et al., 2013). Take QNLI as an example. Given a query Q, we obtain a list of candidate answers $\mathcal{A}$ which contains a positive example A^+ that includes the correct answer,

[4] In this study we use the pre-trained BERT models released by the authors.

Algorithm 1: Training a MT-DNN model.

Initialize model parameters Θ randomly.
Pre-train the shared layers (i.e., the lexicon encoder and the transformer encoder).
Set the max number of epoch: $epoch_{max}$.
//Prepare the data for T tasks.
for t *in* $1, 2, ..., T$ **do**
  Pack the dataset t into mini-batch: D_t.
end
for $epoch$ *in* $1, 2, ..., epoch_{max}$ **do**
  1. Merge all the datasets:
    $D = D_1 \cup D_2 ... \cup D_T$
  2. Shuffle D
  for b_t *in* D **do**
    //b_t is a mini-batch of task t.
    3. Compute loss : $L(\Theta)$
      $L(\Theta)$ = Eq. 6 for classification
      $L(\Theta)$ = Eq. 7 for regression
      $L(\Theta)$ = Eq. 8 for ranking
    4. Compute gradient: $\nabla(\Theta)$
    5. Update model: $\Theta = \Theta - \epsilon\nabla(\Theta)$
  end
end

and $|\mathcal{A}| - 1$ negative examples. We then minimize the negative log likelihood of the positive example given queries across the training data

$$-\sum_{(Q,A^+)} P_r(A^+|Q), \quad (8)$$

$$P_r(A^+|Q) = \frac{\exp(\gamma \text{Rel}(Q, A^+))}{\sum_{A' \in \mathcal{A}} \exp(\gamma \text{Rel}(Q, A'))}, \quad (9)$$

where Rel(.) is defined by Equation 5 and γ is a tuning factor determined on held-out data. In our experiment, we simply set γ to 1.

4 Experiments

We evaluate the proposed MT-DNN on three popular NLU benchmarks: GLUE (Wang et al., 2018), SNLI (Bowman et al., 2015b), and SciTail (Khot et al., 2018). We compare MT-DNN with existing state-of-the-art models including BERT and demonstrate the effectiveness of MTL with and without model fine-tuning using GLUE and domain adaptation using both SNLI and SciTail.

4.1 Datasets

This section briefly describes the GLUE, SNLI, and SciTail datasets, as summarized in Table 1.

GLUE The General Language Understanding Evaluation (GLUE) benchmark is a collection of nine NLU tasks as in Table 1, including question answering, sentiment analysis, text similarity and textual entailment; it is considered well-designed for evaluating the generalization and robustness of NLU models.

SNLI The Stanford Natural Language Inference (SNLI) dataset contains 570k human annotated sentence pairs, in which the premises are drawn from the captions of the Flickr30 corpus and hypotheses are manually annotated (Bowman et al., 2015b). This is the most widely used entailment dataset for NLI. The dataset is used only for domain adaptation in this study.

SciTail This is a textual entailment dataset derived from a science question answering (SciQ) dataset (Khot et al., 2018). The task involves assessing whether a given premise entails a given hypothesis. In contrast to other entailment datasets mentioned previously, the hypotheses in SciTail are created from science questions while the corresponding answer candidates and premises come from relevant web sentences retrieved from a large corpus. As a result, these sentences are linguistically challenging and the lexical similarity of premise and hypothesis is often high, thus making SciTail particularly difficult. The dataset is used only for domain adaptation in this study.

4.2 Implementation details

Our implementation of MT-DNN is based on the PyTorch implementation of BERT[5]. We used Adamax (Kingma and Ba, 2014) as our optimizer with a learning rate of 5e-5 and a batch size of 32 by following Devlin et al. (2018). The maximum number of epochs was set to 5. A linear learning rate decay schedule with warm-up over 0.1 was used, unless stated otherwise. We also set the dropout rate of all the task specific layers as 0.1, except 0.3 for MNLI and 0.05 for CoLa. To avoid the exploding gradient problem, we clipped the gradient norm within 1. All the texts were tokenized using wordpieces, and were chopped to spans no longer than 512 tokens.

4.3 GLUE Main Results

We compare MT-DNN with its variants and a list of state-of-the-art models that have been submitted

[5]https://github.com/huggingface/pytorch-pretrained-BERT

Corpus	Task	#Train	#Dev	#Test	#Label	Metrics
Single-Sentence Classification (GLUE)						
CoLA	Acceptability	8.5k	1k	1k	2	Matthews corr
SST-2	Sentiment	67k	872	1.8k	2	Accuracy
Pairwise Text Classification (GLUE)						
MNLI	NLI	393k	20k	20k	3	Accuracy
RTE	NLI	2.5k	276	3k	2	Accuracy
WNLI	NLI	634	71	146	2	Accuracy
QQP	Paraphrase	364k	40k	391k	2	Accuracy/F1
MRPC	Paraphrase	3.7k	408	1.7k	2	Accuracy/F1
Text Similarity (GLUE)						
STS-B	Similarity	7k	1.5k	1.4k	1	Pearson/Spearman corr
Relevance Ranking (GLUE)						
QNLI	QA/NLI	108k	5.7k	5.7k	2	Accuracy
Pairwise Text Classification						
SNLI	NLI	549k	9.8k	9.8k	3	Accuracy
SciTail	NLI	23.5k	1.3k	2.1k	2	Accuracy

Table 1: Summary of the three benchmarks: GLUE, SNLI and SciTail.

Model	CoLA 8.5k	SST-2 67k	MRPC 3.7k	STS-B 7k	QQP 364k	MNLI-m/mm 393k	QNLI 108k	RTE 2.5k	WNLI 634	AX	Score
BiLSTM+ELMo+Attn [1]	36.0	90.4	84.9/77.9	75.1/73.3	64.8/84.7	76.4/76.1	-	56.8	65.1	26.5	70.5
Singletask Pretrain Transformer [2]	45.4	91.3	82.3/75.7	82.0/80.0	70.3/88.5	82.1/81.4	-	56.0	53.4	29.8	72.8
GPT on STILTs [3]	47.2	93.1	87.7/83.7	85.3/84.8	70.1/88.1	80.8/80.6	-	69.1	65.1	29.4	76.9
$\text{BERT}^{4}_{\text{LARGE}}$	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7/85.9	92.7	70.1	65.1	39.6	80.5
$\text{MT-DNN}_{\text{no-fine-tune}}$	58.9	94.6	**90.1/86.4**	89.5/88.8	**72.7/89.6**	86.5/85.8	**93.1**	79.1	65.1	39.4	81.7
MT-DNN	**62.5**	**95.6**	**91.1/88.2**	**89.5/88.8**	**72.7/89.6**	**86.7/86.0**	**93.1**	**81.4**	65.1	**40.3**	**82.7**
Human Performance	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	95.9	-	87.1

Table 2: GLUE test set results scored using the GLUE evaluation server. The number below each task denotes the number of training examples. The state-of-the-art results are in **bold**, and the results on par with or pass human performance are in **bold**. MT-DNN uses $\text{BERT}_{\text{LARGE}}$ to initialize its shared layers. All the results are obtained from https://gluebenchmark.com/leaderboard on February 25, 2019. Model references: [1]:(Wang et al., 2018) ; [2]:(Radford et al., 2018); [3]: (Phang et al., 2018); [4]:(Devlin et al., 2018).

Model	MNLI-m/mm	QQP	RTE	QNLI (v1/v2)	MRPC	CoLa	SST-2	STS-B
$\text{BERT}_{\text{LARGE}}$	86.3/86.2	91.1/88.0	71.1	90.5/92.4	89.5/85.8	61.8	93.5	89.6/89.3
ST-DNN	86.6/86.3	91.3/88.4	72.0	96.1/-	89.7/86.4	-	-	-
MT-DNN	**87.1/86.7**	**91.9/89.2**	**83.4**	**97.4/92.9**	**91.0/87.5**	**63.5**	**94.3**	**90.7/90.6**

Table 3: GLUE dev set results. The best result on each task is in **bold**. The Single-Task DNN (ST-DNN) uses the same model architecture as MT-DNN. But its shared layers are the pre-trainedBERT model without being refined via MTL. We fine-tuned ST-DNN for each GLUE task using task-specific data. There have been two versions of the QNLI dataset. V1 is expired on January 30, 2019. The current version is v2. MT-DNN use $\text{BERT}_{\text{LARGE}}$ as their initial shared layers.

to the GLUE leaderboard. The results are shown in Tables 2 and 3.

$\text{BERT}_{\text{LARGE}}$ This is the large BERT model released by the authors, which we used as a baseline. We fine-tuned the model for each GLUE task on task-specific data.

MT-DNN This is the proposed model described in Section 3. We used the pre-trained $\text{BERT}_{\text{LARGE}}$

to initialize its shared layers, refined the model via MTL on all GLUE tasks, and fine-tuned the model for each GLUE task using task-specific data. The test results in Table 2 show that MT-DNN outperforms all existing systems on all tasks, except WNLI, creating new state-of-the-art results on eight GLUE tasks and pushing the benchmark to 82.7%, which amounts to 2.2% absolution improvement over $\text{BERT}_{\text{LARGE}}$. Since MT-DNN uses $\text{BERT}_{\text{LARGE}}$ to initialize its shared layers, the gain is mainly attributed to the use of MTL in refining the shared layers. MTL is particularly useful for the tasks with little in-domain training data. As we observe in the table, on the same type of tasks, the improvements over BERT are much more substantial for the tasks with less in-domain training data than those with more in-domain labels, even though they belong to the same task type, e.g., the two NLI tasks: RTE vs. MNLI, and the two paraphrase tasks: MRPC vs. QQP.

$\text{MT-DNN}_{\text{no-fine-tune}}$ Since the MTL of MT-DNN uses all GLUE tasks, it is possible to directly apply MT-DNN to each GLUE task without fine-tuning. The results in Table 2 show that $\text{MT-DNN}_{\text{no-fine-tune}}$ still outperforms $\text{BERT}_{\text{LARGE}}$ consistently among all tasks but CoLA. Our analysis shows that CoLA is a challenge task with much smaller in-domain data than other tasks, and its task definition and dataset are unique among all GLUE tasks, making it difficult to benefit from the knowledge learned from other tasks. As a result, MTL tends to underfit the CoLA dataset. In such a case, fine-tuning is necessary to boost the performance. As shown in Table 2, the accuracy improves from 58.9% to 62.5% after fine-tuning, even though only a very small amount of in-domain data is available for adaptation. This, together with the fact that the fine-tuned MT-DNN significantly outperforms the fine-tuned $\text{BERT}_{\text{LARGE}}$ on CoLA (62.5% vs. 60.5%), reveals that the learned MT-DNN representation allows much more effective domain adaptation than the pre-trained BERT representation. We will revisit this topic with more experiments in Section 4.4.

The gain of MT-DNN is also attributed to its flexible modeling framework which allows us to incorporate the task-specific model structures and training methods which have been developed in the single-task setting, effectively leveraging the existing body of research. Two such examples are the use of the SAN answer module for the pairwise text classification output module and the pairwise ranking loss for the QNLI task which by design is a binary classification problem in GLUE. To investigate the relative contributions of these modeling design choices, we implement a variant of MT-DNN as described below.

ST-DNN ST-DNN stands for Single-Task DNN. It uses the same model architecture as MT-DNN. But its shared layers are the pre-trained BERT model without being refined via MTL. We then fine-tuned ST-DNN for each GLUE task using task-specific data. Thus, for pairwise text classification tasks, the only difference between their ST-DNNs and BERT models is the design of the task-specific output module. The results in Table 3 show that on all four tasks (MNLI, QQP, RTE and MRPC) ST-DNN outperforms BERT, justifying the effectiveness of the SAN answer module. We also compare the results of ST-DNN and BERT on QNLI. While ST-DNN is fine-tuned using the pairwise ranking loss, BERT views QNLI as binary classification and is fine-tuned using the cross entropy loss. ST-DNN significantly outperforms BERT demonstrates clearly the importance of problem formulation.

4.4 Domain Adaptation Results on SNLI and SciTail

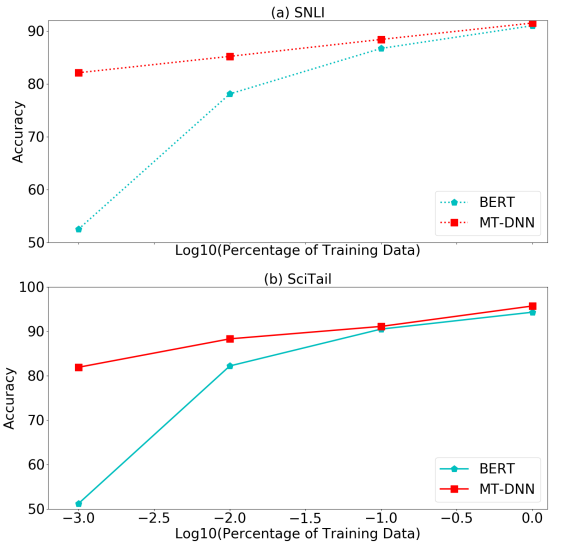

Figure 2: Domain adaption results on SNLI and SciTail development datasets using the shared embeddings generated by MT-DNN and BERT, respectively. Both MT-DNN and BERT are fine-tuned based on the pre-trained $\text{BERT}_{\text{BASE}}$. The X-axis indicates the amount of domain-specific labeled samples used for adaptation.

Model	0.1%	1%	10%	100%
SNLI Dataset (Dev Accuracy%)				
#Training Data	549	5,493	54,936	549,367
BERT	52.5	78.1	86.7	91.0
MT-DNN	82.1	85.2	88.4	91.5
SciTail Dataset (Dev Accuracy%)				
#Training Data	23	235	2,359	23,596
BERT	51.2	82.2	90.5	94.3
MT-DNN	81.9	88.3	91.1	95.7

Table 4: Domain adaptation results on SNLI and SciTail, as shown in Figure 2.

One of the most important criteria of building practical systems is fast adaptation to new tasks and domains. This is because it is prohibitively expensive to collect labeled training data for new domains or tasks. Very often, we only have very small training data or even no training data.

To evaluate the models using the above criterion, we perform domain adaptation experiments on two NLI tasks, SNLI and SciTail, using the following procedure:

1. use the MT-DNN model or the BERT as initial model including both *BASE* and *LARGE* model settings;

2. create for each new task (SNLI or SciTail) a task-specific model, by adapting the trained MT-DNN using task-specific training data;

3. evaluate the models using task-specific test data.

We starts with the default training/dev/test set of these tasks. But we randomly sample 0.1%, 1%, 10% and 100% of its training data. As a result, we obtain four sets of training data for SciTail, which respectively includes 23, 235, 2.3k and 23.5k training samples. Similarly, we obtain four sets of training data for SNLI, which respectively include 549, 5.5k, 54.9k and 549.3k training samples.

We perform random sampling five times and report the mean among all the runs. Results on different amounts of training data from SNLI and SciTail are reported in Figure 2. We observe that MT-DNN outperforms the BERT baseline consistently with more details provided in Table 4. The fewer training examples used, the larger improvement MT-DNN demonstrates over BERT. For example, with only 0.1% (23 samples) of the SNLI training data, MT-DNN achieves 82.1% in accuracy while BERT's accuracy is 52.5%; with 1% of the training data, the accuracy from MT-DNN is 85.2% and BERT is 78.1%. We observe similar results on SciTail. The results indicate that the representations learned by MT-DNN are more consistently effective for domain adaptation than BERT.

In Table 5, we compare our adapted models, using all in-domain training samples, against several strong baselines including the best results reported in the leaderboards. We see that MT-DNN$_{\text{LARGE}}$ generates new state-of-the-art results on both datasets, pushing the benchmarks to 91.6% on SNLI (1.5% absolute improvement) and 95.0% on SciTail (6.7% absolute improvement), respectively. This results in the new state-of-the-art for both SNLI and SciTail. All of these demonstrate the exceptional performance of MT-DNN on domain adaptation.

Model	Dev	Test
SNLI Dataset (Accuracy%)		
GPT (Radford et al., 2018)	-	89.9
Kim et al. (2018)*	-	90.1
BERT$_{\text{BASE}}$	91.0	90.8
MT-DNN$_{\text{BASE}}$	91.5	91.1
BERT$_{\text{LARGE}}$	91.7	91.0
MT-DNN$_{\text{LARGE}}$	**92.2**	**91.6**
SciTail Dataset (Accuracy%)		
GPT (Radford et al., 2018)*	-	88.3
BERT$_{\text{BASE}}$	94.3	92.0
MT-DNN$_{\text{BASE}}$	95.7	94.1
BERT$_{\text{LARGE}}$	95.7	94.4
MT-DNN$_{\text{LARGE}}$	**96.3**	**95.0**

Table 5: Results on the SNLI and SciTail dataset. Previous state-of-the-art results are marked by $*$, obtained from the official SNLI leaderboard (https://nlp.stanford.edu/projects/snli/) and the official SciTail leaderboard maintained by AI2 (https://leaderboard.allenai.org/scitail).

5 Conclusion

In this work we proposed a model called MT-DNN to combine multi-task learning and language model pre-training for language representation learning. MT-DNN obtains new state-of-the-art results on ten NLU tasks across three popular benchmarks: SNLI, SciTail, and GLUE. MT-DNN also demonstrates an exceptional generalization capability in domain adaptation experiments.

There are many future areas to explore to improve MT-DNN, including a deeper understanding of model structure sharing in MTL, a more effective training method that leverages relatedness among multiple tasks, for both fine-tuning and pre-training (Dong et al., 2019), and ways of incorporating the linguistic structure of text in a more explicit and controllable manner. At last, we also would like to verify whether MT-DNN is resilience against adversarial attacks (Glockner et al., 2018; Talman and Chatzikyriakidis, 2018; Liu et al., 2019).

Acknowledgments

We would like to thanks Jade Huang from Microsoft for her generous help on this work.

References

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

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015b. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.

J. Gao, M. Galley, and L. Li. 2018. Neural approaches to conversational AI. *CoRR*, abs/1809.08267.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *The 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Soft layer-specific multi-task summarization with entailment and question generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 687–697.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTail: A textual entailment dataset from science question answering. In *AAAI*.

Seonhoon Kim, Jin-Hyuk Hong, Inho Kang, and Nojun Kwak. 2018. Semantic sentence matching with densely-connected recurrent and co-attentive information. *arXiv preprint arXiv:1805.11360*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xiaodong Liu, Kevin Duh, and Jianfeng Gao. 2018a. Stochastic answer networks for natural language inference. *arXiv preprint arXiv:1804.07888*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2018b. Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. pages 2383–2392.

Sebastian Ruder12, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2019. Latent multi-task architecture learning.

Aarne Talman and Stergios Chatzikyriakidis. 2018. Testing the generalization power of neural network models across nli benchmarks. *arXiv preprint arXiv:1810.09774*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Yichong Xu, Xiaodong Liu, Yelong Shen, Jingjing Liu, and Jianfeng Gao. 2018. Multi-task learning for machine reading comprehension. *arXiv preprint arXiv:1809.06963*.

Yu Zhang and Qiang Yang. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*.