Multi-Task Convolutional Neural Network for Text Classification

Liqiang Xiao
Shanghai Jiao Tong University
xiaoliqiang@sjtu.edu.cn

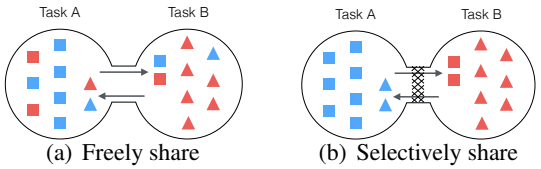
Abstract

Multi-task learning has been proven effective on many NLP problems, which focuses on sharing knowledge among tasks through shared layers. Recent works are prone to share deeper layers among tasks to exchange advanced knowledge. However, most existing approaches attend to mix the whole feature maps proportionally but have no selection for features. In this way the feature space for current task may be easily contaminated by helpless features borrowed from others, which inevitably suffer from the interference between tasks. In this paper, we propose a multi-task convolutional neural network that is able to share the features among tasks in a selective way. This advantage is achieved by two cross-task modules that can filter the helpful features. Experiments on six different tasks for text classification validate the benefits of our approaches. Furthermore, we also investigate some factors that may affect the performance of our multi-task CNN model.

1 Introduction

CNN models have achieved impressive results on many natural language processing tasks, which use convolving filters to extract local features and have been successfully applied in computer vision field. In recent years, increasing works transfer CNNs to natural language processing tasks, such as representation learning (Liu et al., 2015), information retrieval (Shen et al., 2014), text classification (Kalchbrenner et al., 2014), etc. In particular, feed-forward CNNs with word embedding have been proven to be a relatively simple yet powerful kind of models for text classification (Kim, 2014).

However, the reliance on large-scale corpus has been a formidable constraint for deep neural networks (DNNs) based methods due to the numerous parameters. It costs a lot for a large-scale dataset due to the huge volume of parameters, because constructing a large-scale labeled dataset is extremely labor-intensive. To solve this problem, these models usually employ a pre-trained phase to map words into vectors with semantic implication (Collobert et al., 2011), which just introduces extra knowledge and does not directly optimize the target task. The problem of insufficient annotated resources is not intrinsically solved either.



(a) Freely share (b) Selectively share

Figure 1: Two schemes for sharing knowledge among tasks. Boxes and triangles denote the features for Task A and B respectively. The red boxes and blue triangles represent the features can be shared to benefit the other task.

Multi-task learning (MTL) can implicitly increase the corpus size and create a synergy effect among datasets (Caruana, 1997). By learning related tasks in parallel, MTL has the ability to exploit the relations between similar tasks to benefit each other, and eventually improves the performance for classification. In addition, models handling multiple tasks also benefit from a regularization effect to decrease the overfitting. It can help the models to learn a more universal representation for text sequences. Inspired by this, more DNN-based models (Collobert and Weston, 2008; Liu et al., 2015, 2016) utilize multi-task learning to improve their performance. Traditionally, multi-task learning only shares a few shallow layers among tasks, so only low-level knowledge is exchanged to benefit each other (Collobert and Weston, 2008). But recently more works prefer to share deeper layers to share more high-level

knowledge, which has been proven effective to enhance the performance (Liu et al., 2015) (Liu et al., 2016).

However, most existing works share the knowledge through adding the tasks' feature maps proportionally (Ruder et al., 2017) (Misra et al., 2016), sharing the features with the same weight. In this way, as shown in Fig.1(a), some helpless features may be borrowed from others, which would mislead the predictions or burden the network for distinguish the helpful features. Moreover, most existing models assign the same subnets to each task. These subnets learn the inputs in similar way and restrict the richness of perspective that has been proven effective in text representation (He et al., 2015). With highly similar feature spaces, the reference value for features is reduced among tasks, which lower the benefits of multi-task learning.

To resolve above two problems, we propose a new CNN-based models for multi-task learning, which can share the features in a selective way (Fig.1(b)) and increase the feature variety between tasks by employing different subnets for tasks.

Our model employs a relatively separative structure, in which the tasks can share the knowledge in a selective way through two new modules, namely, *cross-task convolution* and *cross-task pooling*. So there is less interference in our model for its elaborate structure.

We conduct extensive experiments on six benchmark datasets for text classification. And the result shows that our multi-task model gains great performance against the single-task CNNs and other competitors.

The contributions of this paper can be briefly summarized as follows:

- We propose a new structure for multi-task learning, which can reduce the undesirable interferences among tasks. And their thoughts can be easily referred to by other works.
- By design the subnet for task in different way, our model augments the feature variety among different tasks. It augments the reference values of feature spaces each others.
- Our model demonstrates strong results on several benchmark classification datasets and outperforms the state-of-the-art baselines on 4 datasets.

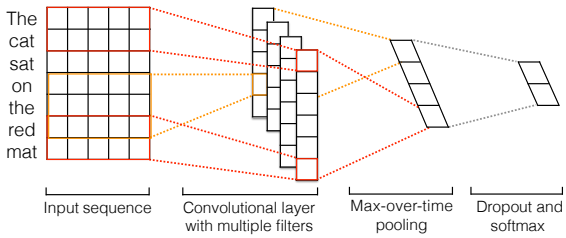


Figure 2: A single-task CNN models for text classification.

2 CNN for Text Classification

The main capability of DNNs for text classification is to represent word sequences into fix-length vectors. There are many frameworks can be used for sentence modeling, involving Neural Bag-of-Words (NBOW) model, recursive neural network (RecNN) (Socher et al., 2012, 2013), recurrent neural network (RNN) (Chung et al., 2014) and convolutional neural network (CNN) (Collobert et al., 2011).

In recent years, CNN has shown its advantages in the NLP field. CNN models can not only handle input sentences of varying length but also capture short and long range relations through the feature graph over the sentence (Kalchbrenner et al., 2014). The CNN architecture illustrated in Fig. 2 is a typical one-dimension convolutional network. In this paper, this kind of CNN architecture is defined as the subnet in our multi-task model, which can be detailed as follows.

2.1 Convolution Neural Network

Given an text sequence $x_{1:n} = x_1 x_2 \cdots x_l$, we first use a lookup table to get the embedding results (word vector) $\mathbf{x}_i$ for each word x_i. Then the input matrix can be represented as $\mathbf{x} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \cdots \oplus \mathbf{x}_l$ by concatenating each word vector, where $\oplus$ denotes the operation of concatenation. CNNs produce the representation of the input sequence through staking the layer of convolution, pooling in order, which can be briefly formalized as:

$$\boldsymbol{F} = \mathbf{H} * \mathbf{x} \tag{1}$$

$$\hat{\boldsymbol{F}} = pooling(\boldsymbol{F}) \tag{2}$$

$$\mathbf{y} = \boldsymbol{w}\hat{\boldsymbol{F}} + \boldsymbol{b}, \tag{3}$$

where $\boldsymbol{H}$ is the filter for convolutional operation $*$; $pooling$ denotes the pooling operation; $\boldsymbol{F}$ and

$\hat{\boldsymbol{F}}$ represent the feature maps. $\boldsymbol{w}$ and $\boldsymbol{b}$ denote the weight and bias perspectively in fully connected layer. Usually, drop out mechanism is used as a kind of regularization for output layer and the Eq.3 can be rewrited as

$$\mathbf{y} = \boldsymbol{w}\hat{\boldsymbol{F}} \circ \boldsymbol{r} + \boldsymbol{b}. \tag{4}$$

Here $\circ$ refers to the element-wise multiplication operator, and $\mathbf{r} \in \mathbb{R}^d$ is a "masking" vector comprising Bernoulli random variables with probability p of being 1.

2.2 Output Layer for Text Classification

Following the pooling layer, a fully connected layer with dropout and the softmax layer was used, which transforms the vector representation into the probability distribution over classes.

During backpropagation, the parameters in the network are updated by gradient of the loss between the predicted and true distributions. Such as cross-entropy function

$$L(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i=1}^{N}\sum_{j=1}^{C} \mathbf{y}_i^j log(\hat{\mathbf{y}}_i^j), \tag{5}$$

where $\hat{\mathbf{y}}$ is the predicted probability distribution; $\mathbf{y}$ denotes the ground-truth label; N and C are the batch size and the number of classes respectively.

3 Multi-Task CNN Model

The key factor for multi-task learning is the scheme for sharing the features. So an elaborate architecture that can well control the features sharing among tasks is very crucial for multi-task learning. A good architecture should not only abundantly exchange the features to help tasks extend its feature space, but also select the helpful features and avoid the contamination. So, in this section, we propose a selective architecture to optimize the sharing scheme.

3.1 Model Architecture

Multi-task model with deeper layers shared can fuse high-level knowledge and greatly increase the feature space. But undesirable interference is inevitable and simultaneously comes with the benefits, especially between the less-related tasks. This would burden the models with the overhead on distinguishing helpful features. To overcome the above problem, we explore another scheme as Fig. 1(b), in which the tasks have ability to selectively borrow helpful knowledge from others.

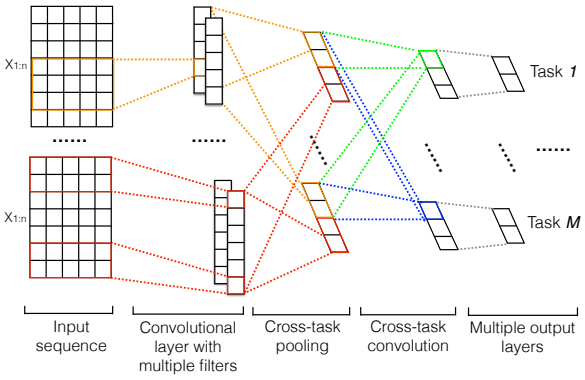


Figure 3: Illustration of two multi-task CNN architectures for modeling text.

To reduce interference, we assign each task a private subnet, which is illustrated in Fig.3. Tasks are relatively separated and can borrow the useful information from others through the bridges: *Cross-Task Pooling* and *Cross-Task Convolution*. We set parameters in these two modules to control the feature sharing. The weights are updated through backpropagation to optimize the selection and need no more supervision. This is an end-to-end and easy-training method and the thoughts are ready to be employed by other works. The details will be introduced in section 3.2 and 3.3.

To increase the variety of feature space and create more perspectives, the filters assigned to different pipelines are of different sizes. We also use two kinds of pooling methods in cross-task pooling layer.

3.2 Cross-Task Pooling

In our model, convolutional layer captures features with the filters of various sizes. For filters in different tasks, here are two distinctions between them. One is that the sizes of them are different, which deal with the input in different perspectives. Another is that each dataset have its unique characteristics so that tasks update the network in distinct ways and form different knowledge for feature extraction (convolutional layers) and feature selection (pooling layers). To let tasks borrow knowledge from others, every time we feed one task into all the pipelines, then, exchange the obtained features through a new module called cross-task pooling (CT-Pooling).

In CT-Pooling module, max-pooling operation

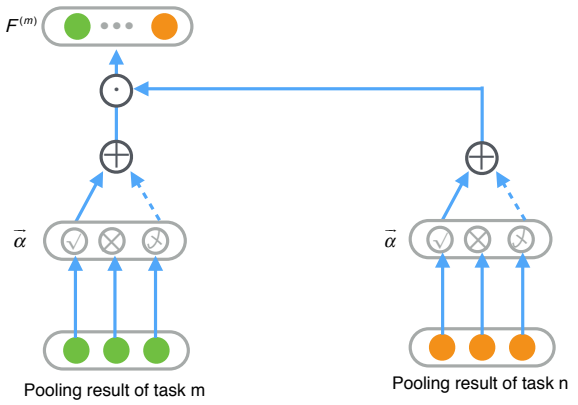


Figure 4: Illustration of the mechanism of parameter $\boldsymbol{\alpha}$ in cross-task pooling.

in each task produces their own feature maps $\mathbf{F}^{(m)}_{max}$ by concatenating the max item of every $\mathbf{F}$. We extend feature map in hand by other tasks' mean-pooling result $\mathbf{F}^{(m)}_{mean}$. The newly formed feature maps will be

$$\begin{aligned}\mathbf{F}^{(m)} =& \boldsymbol{\alpha}^{(1)}\mathbf{F}^{(1)}_{mean} \oplus \boldsymbol{\alpha}^{(2)}\mathbf{F}^{(2)}_{mean} \oplus \cdots \\ & \oplus \boldsymbol{\alpha}^{(m)}\mathbf{F}^{(m)}_{max} \oplus \cdots \oplus \boldsymbol{\alpha}^{(M)}\mathbf{F}^{(M)}_{mean},\end{aligned} \tag{6}$$

where $\oplus$ is the concatenation operation and M is the number of the subnet. $\boldsymbol{\alpha}^{(m)}$ is an array to adjust the weight of the feature merging into current task and its value are auto optimized through backpropagation. It is in the same size with feature map, so that each item in it corresponds to one feature and control its weight. Fig. 4 illustrates the mechanism how parameter $\boldsymbol{\alpha}$ filter the pooling results and form the new feature map $\mathbf{F}^{(m)}$.

Cross-task pooling operation not only promotes knowledge exchange among subnets, but also further enriches the diversity of features through introducing another pooling method. So far, our neural network has contained rich features. The produced multiple feature maps will be fed into cross-task convolution layer to distill more global features.

3.3 Cross-Task Convolution

Traditional way to merge the features of tasks is adding the whole feature maps up proportionally (Misra et al., 2016) (Ruder et al., 2017), so the weights of features are equal in the same feature map. This kind of weights just represents the cohesion between task and have no consideration for feature level. In another word, they select only the strong-related tasks but have no selection for helpful features. To better fuse the features cross the tasks, we design another module named cross-task convolution (CT-Conv), which can conduct selection in feature level.

Let $\mathbf{F}^{(m)}$ refer to the feature map of task m, which is generated in CT-Pooling layer. For all M tasks, multiple feature maps can be represented as $\mathbb{F} : \{\mathbf{F}^{(1)}, \cdots, \mathbf{F}^{(m)}, \ldots \mathbf{F}^{(M)}\}$ will be emitted in parallel. Each feature map $\hat{\mathbf{F}}^{(m)}$ in the next layer is computed by convolving a distinct set of filters arranged in a matrix $\mathbf{m}^{(m)}_k$ with each feature map $\mathbf{F}^{(k)}$ in CT-Pooling and summing the results:

$$\hat{\mathbf{F}}^{(m)} = \sum_{k=1}^{M} \boldsymbol{\beta}^{(k)}\mathbf{m}^{(m)}_k * \mathbf{F}^{(k)}, \tag{7}$$

where $*$ is a series of convolution operations and M is the number of the tasks. $\boldsymbol{\beta}^{(k)}$ in this place functions similarly with parameter $\boldsymbol{\alpha}$. Both of them control the weights for passing features and select out the helpful ones, and their values are auto optimized by backpropagation.

3.4 Multi-Perspective

Although the selective modules can filter out the helpful features from other tasks to enhance the current one, there is no guarantee that there is plenty of helpful features if the feature spaces for different tasks are highly similar. To give an extreme example, if the feature spaces for different tasks are all the same due to the same structure for subnets, there is no new feature can be borrowed from other tasks and synergy cannot be generated too.

Thus, to increase the variety of feature space, we increase the perspective for tasks in two aspects—filter length and pooling method. For the former, we emit variety for features by assigning tasks different length of filters, in which way feature lengths for tasks are different. For the pooling, we the extract the most import features for current task by max-pooling, and others by mean-pooling. With more diversity, feature spaces for tasks have more change to complement and extend each other, which provide more resources to help the model generate right predictions.

3.5 Training

In the last layer of model, vector representations of input sequences have to be fed into different output layers to fit the number of class, which emits

Dataset	Target	Type	Train Size	Dev. Size	Test Size	Class	Avg. Length
SST-1	Sentiment	Sentence	8544	1101	2210	5	19
SST-2		Sentence	6920	872	1821	2	19
IMDB		Document	25000	-	25000	2	279
SUBJ	Subjectivity	Sentence	9000	-	1000	2	21
RN	Topics	Document	8964	-	2239	46	146
QC	Question Types	Sentence	5153	-	489	6	10

Table 1: Statistics of the six text classification datasets. Dev. and Avg. are the abbreviations of development and average respectively.

the prediction of probability distribution for the task m

$$\hat{\mathbf{y}}^{(m)} = softmax(\mathbf{W}^{(m)}\hat{\mathbf{F}}^{(m)} + \mathbf{b}^{(m)}), \quad (8)$$

where $\hat{\mathbf{y}}^{(m)}$ is predictive result, $\mathbf{W}^{(m)}$ is the weight of the full-connected layer, and $\mathbf{b}^{(m)}$ is the bias term.

Given the prediction of all tasks, a global loss function forces models to take every task into account.

$$\Phi = \sum_{m=1}^{M} \lambda_m L(\hat{\mathbf{y}}^{(m)}, \mathbf{y}^{(m)}), \quad (9)$$

where λ_m is the weight for the task m. In this paper, we simply set λ_m to $1/M$ for all M tasks to make a balance.

In order to train the parameters with different datasets, following (Collobert and Weston, 2008), each task is trained by turn in a stochastic manner. The steps can be described as follows:

1. Pick up a task m randomly;
2. Select an arbitrary sample s from the task m;
3. Feed the sample s into the model and update the parameters;
4. Go back to 1.

Following the training phase, we employ fine tuning strategy (Liu et al., 2016) to further optimize the parameters for each task alone.

4 Experiments

In this section, we investigate the empirical performances of our models on six related benchmark tasks for text classification. And the results are compared with the state-of-the-art models. Furthermore, we study some factors that have direct impact on performance.

4.1 Datasets

As Table 1 shows, we select six benchmark datasets for text classification that are related to each other to different degree. These datasets contain sentiment, subjectiveness, topic and question type classification, which belong to different class of target and can be briefly introduced as follows:

- **SST-1** Stanford Sentiment Treebank[1] (Socher et al., 2013), a movie review dataset with five classes of labels (very positive, positive, neutral, negative, very negative), and split into train/dev/test three parts.
- **SST-2** Also from the Stanford Sentiment Treebank, but with binary labels.
- **SUBJ** Subjectivity dataset[2] that the task is to classify a sentence level text as being subjective or objective (Pang et al., 2004).
- **IMDB** Binary class dataset[3] (Maas et al., 2011) consisting of 100,000 movie reviews that are document-level.
- **RN** Reuters Newswire topics classification (Velasco et al., 1994).
- **QC** Question dataset classifying a given question into six types (Li and Roth, 2002).

4.2 Hyperparameters and Training

Initializing word vectors with the dataset trained by an unsupervised neural network model is an efficient method to enhance the performance without a large-scale supervised training data (Collobert et al., 2011; Socher et al., 2011; Iyyer et al.,

[1] http://nlp.stanford.edu/sentiment.

[2] http://www.cs.cornell.edu/people/pabo/movie-review-data/

[3] http://ai.stanford.edu/ amaas/data/ sentiment/

[4] subnets in the first convolutional layers use the filters of size 2,3,4,5,6,7 respectively; Cross-task convolution only uses the filters of size 3.

Hyperparameter	Setting
Embedding size	300
Dropout rate	0.5
Mini-batch size	30
Learning rate	0.001
l_2 constraint	0.1
Filter size[6]	2,3,4,5,6,7 | 3
Filter number	50×6

Table 2: Hyperparameter settings

Model	Single-Task	Model
SST-1	45.8	49.7 (+3.9)
SST-2	85.6	88.5 (+2.9)
IMDB	88.6	91.3 (+2.7)
SUJB	91.7	94.5 (+2.8)
RN	82.5	84.5 (+2.0)
QC	88.6	93.8 (+5.2)
AvgΔ	-	**+3.3**

Table 3: Performance of our models against single-task. The number in bracket denotes the improvement.

2014). In all of our experiments, our models contain a lookup table by employing Word2Vec (Mikolov et al., 2013) trained on Google News, which comprises more than 100B words with a vocabulary size of around 3M. Word2Vec derives from a continuous bag-of-words architecture and each vector has 300 dimensions.

Word vectors are further fine-tuned during training to get a more optimized embedding result that fits the datasets. The whole network is trained through stochastic gradient decent using Adadelta update rule (Zeiler, 2012). For datasets without development set, we randomly select 10% of the training data as the dev set. Final hyperparameter settings illustrate in Table 2.

4.3 Performance of Multi-task CNN

We simultaneously train our model on six datasets and compare it with single-task scenario as Fig.2. As the result shown in Table 3, our models give an improvement over the single task, which demonstrates the positive synergy and regularization effect of multi-task, especially between tasks subjecting to the same target. Model shows its advantage beyond Model-I in architecture making a balance in task-specific and universal characteristics. For instance, RN and QC dataset that belongs to distinct targets improve less in normal model, which may be caused by interference between less-related tasks.

Our model assigns a private subnet to each task, including lookup table. It selectively allows the information flow across the subnets through cross-task modules. Pre-trained vectors are also used on both single-task and multi-task cases. The average improvement of 3.3% demonstrates the effectiveness of its separative architecture.

4.4 Comparisons with the State-of-the-art Models

We compare our proposed model against the following models:

- **RNTN** Recursive Neural Tensor Network with tensor-based feature function and parse trees (Socher et al., 2013).
- **DCNN** Convolutional Neural Network with a novel dynamic k-max pooling (Kalchbrenner et al., 2014)
- **MGNC-CNN** Multi-group norm constraint CNN. We use the result of MGNC-CNN(w2v+Syn+Glv) (Zhang et al., 2016).
- **PV** Paragraph vectors based logistic regression (Le and Mikolov, 2014).
- **MT-GRNN** A general multi-task learning architecture with Recurrent Neural Network (Zhang et al., 2017).
- **MT-RNN** Multi-task learning with Recurrent Neural Networks by a shared-layer architecture (Liu et al., 2016)
- **MT-DNN** Multi-Task learning with Deep Neural Networks that uses bag-of-word representations and a hidden shared layer (Liu et al., 2015).
- **MT-CNN** Multi-Task learning with Convolutional Neural Network that partially shares a lookup table (Collobert and Weston, 2008).

As shown in Table 4, our model outperforms the state-of-the-art works in four datasets and show competitive results in the other one. Competitors slightly outperform our models in IMDB, since its unique character that the sequences are much longer than other datasets. The results demonstrate that our strategies for selecting features and increasing variety are effective for multi-task learning.

Model	SST-1	SST-2	IMDB	SUBJ	QC
RNTN (Socher et al., 2013)	45.7	85.4	-	-	-
DCNN (Kalchbrenner et al., 2014)	48.5	86.8	-	-	-
MGNC-CNN (Zhang et al., 2016)	48.7	88.3	-	94.1	-
PV (Le and Mikolov, 2014)	44.6	82.7	**91.7**	90.5	91.8
MT-GRNN(Zhang et al., 2017)	49.2	87.7	91.6	94.0	92.3
MT-RNN (Liu et al., 2016)	49.6	87.9	91.3	94.1	-
MT-CNN(Collobert and Weston, 2008)	49.0	86.9	91.5	94.1	92.0
MT-DNN(Liu et al., 2015)	48.1	87.3	91.4	94.0	92.1
Our model	**49.7**	**88.5**	91.3	**94.5**	**93.8**

Table 4: Results of our model against other state-of-the-art models.

Models	IMDB	SST-2
Normal MT-CNN	good, great, nice, terrific, bad, lousy pointless, well-directed,	good, great, nice, terrific, bad, lousy, pointless, decent
Our model	good, great, decent, well-directed, solid , cheep, terrific, poor	good, great, nice, decent, terrific, bad, lousy, pointless, stupid

Table 5: Visualization of typical features.

4.5 Visualization

To intuitively illustrate the feature spaces of tasks and how our model shares them, in this section we intuitively visualize the features that our model captures. We employ a normal multi-task CNN as the baseline, which shares all the layers for tasks except output layer. In the first convolutional layer, a filter is connected with a neuron in the output feature map. When connected filter capture the crucial N-gram, the neuron will be active. We count the words that the filters obtained and visualize them in Table 5.

We have observed that, for normal MT-CNN, there is a huge intersection of features between two tasks. In another word, the feature spaces for two tasks are highly similar. Thus, there are less feature a tasks can borrow from others, which reduces the value of reference of these feature spaces. On the contrary, the feature spaces in our model have more diversity. Through mutual complement, feature spaces can be extended, especially for the features appear less. For example, the word 'poor' appears a lot of times in datasets IMDB but appears much less in SST-2. Thus, it is difficult for the subnet of task SST-2 to capture that feature. However, through cross-task modules, subnet for SST-2 can easily borrow this kind of features from subnet for IMDB, which can help the network make right prediction.

5 Discussion

5.1 Efficacy of Each Cross-Task Module

In the last two subsections, the effectiveness of our models has been adequately verified. Benefiting from two cross-task modules that exchange knowledge among tasks, our model outperforms the others. To validate the independent efficacy of two modules, extra ablation studies are conducted. We test two variants of model on all datasets, which only contain one cross-task module.

The results are illustrated in Fig. 5. As expected, the single-task model shows less competitive performance because of the insufficient training dataset and the small feature space. And two variants using cross-task modules both yield an improvement on the performance. We find that CT-Conv contributes more to three sentiment tasks and CT-Pooling to three others, which have different targets. Our inference is that: 1) features appearing in similar tasks are more likely to be used by each other, and CT-Conv can fuse them to generate more useful high-level features. 2) on the other hand, tasks with different targets have relatively more task-specific features, for which the selection in CT-Pooling is more necessary.

The experiments demonstrate that both modules we proposed boost the performance of the tasks.

5.2 Effect of Task Number

As for multi-task learning, the number of the dataset is a significant factor that influences the performance. So we test our models with series of

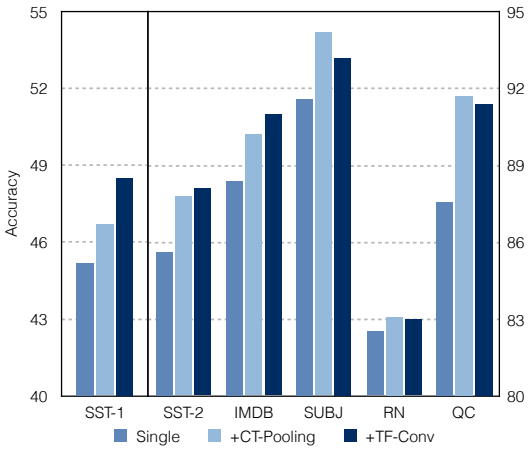


Figure 5: Independent performance of cross-task operations. Single: no cross-task operation; +CT-Pooling: uses only cross-task pooling; +TF-Conv: uses only cross-task convolution.

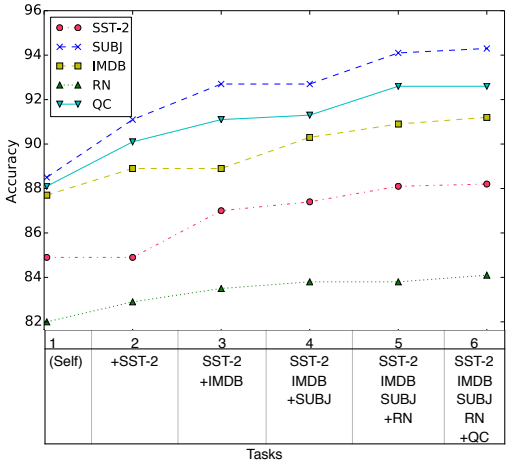


Figure 6: Influence of task number for accuracy. The datasets with '+' are appended currently.

incremental experiments. We fix the first task and sequentially append other tasks into the model. As Fig. 6 shows, the accuracy of the fixed task steadily improves and eventually saturates when more tasks are involved in.

6 Related Work

In the NLP field, NN-based multi-task learning has been proven to be effective (Collobert and Weston, 2008; Liu et al., 2015, 2016). The synergy between multi-task learning and neural network is obvious. The earliest idea can be traced back to (Caruana, 1997).

Collobert and Weston (2008) develops a multi-task learning model based on CNN. It shares only the partial lookup table to train a better word embedding. And Liu et al. (2015) proposes a DNN based model for multi-task learning, which shares some low layers to represent the text. These works allow the tasks to reuse low-level layers but separate the high-level layers.

Some models are proposed to sharing deeper layer of networks, which can exchange high level knowledge among tasks and gain better performance. Liu et al. (2016) and Zhang et al. (2017) introduce some RNN architectures and design different schemes for knowledge sharing among tasks. These trials promote the performance of models, but they give no consideration to the interference in multi-task learning.

Thus, more approaches are proposed to reduce the interference among tasks. One is to strengthen connection between the strong-related tasks and weaken the less-related ones. Misra et al. (2016) proposes a model for images, trying to reduce the interference through weighting the connections between tasks. Though this kind of methods reduces the noise from less-related tasks, the processing is coarse-grained and cannot distinguish the useful information in smaller feature level. In another way, Liu et al. (2017) aims to capture task-invariant information for multiple related tasks using adversarial training strategy. They separate out and only share a common feature space for all the tasks to reduce the noisy features.

Different from these models, our model weights the connections between tasks in feature level, which passes all the useful features for the task in hand. By that our model can reduce interference without extra supervision. And the dissymmetric architecture increases the diversity for feature spaces, which augments the mutual complement.

7 Conclusion and Feature Work

In this paper, we have proposed a CNN based framework for multi-task learning, which has the parameters to control the passing of information in feature level. By passing the helpful features and blocking the useless ones, our model reduce the interference between the tasks. We also design our model in a dissymmetric way to enhance the reference value for each other. Experiments on six datasets demonstrate the effectiveness of our model for text classification. And we also visualize the property of our models and investigate the performance in different scenarios.

References

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

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR* abs/1412.3555. http://arxiv.org/abs/1412.3555.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. pages 160–167. https://doi.org/10.1145/1390156.1390177.

Ronan Collobert, Jason Weston, L Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(1):2493–2537.

Hua He, Kevin Gimpel, and Jimmy J. Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 1576–1586. http://aclweb.org/anthology/D/D15/D15-1181.pdf.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Meeting of the Association for Computational Linguistics*. pages 1113–1122.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*. pages 655–665. http://aclweb.org/anthology/P/P14/P14-1062.pdf.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1746–1751. http://aclweb.org/anthology/D/D14/D14-1181.pdf.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *Computer Science* 4:1188–1196.

Xin Li and Dan Roth. 2002. Learning question classifiers. *Coling* 12(24):556–562.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning .

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. pages 1–10. https://doi.org/10.18653/v1/P17-1001.

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 *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 912–921.

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 *Meeting of the Association for Computational Linguistics: Human Language Technologies*. pages 142–150.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems* 26:3111–3119.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Pang, Bo, Lee, and Lillian. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. *Proceedings of Acl* pages 271–278.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Sgaard. 2017. Sluice networks: Learning what to share between loosely related tasks .

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*. pages 101–110. https://doi.org/10.1145/2661829.2661935.

R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank .

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Joint*

Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. pages 1201–1211.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John Mcintyre Conference Centre, Edinburgh, Uk, A Meeting of Sigdat, A Special Interest Group of the ACL*. pages 151–161.

E Velasco, L. C. Thuler, C. A. Martins, L. M. Dias, and V. M. Gonalves. 1994. Automated learning of decision rules for text categorization. *Acm Transactions on Information Systems* 12(3):233–251.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *Computer Science* .

Honglun Zhang, Liqiang Xiao, Yongkun Wang, and Yaohui Jin. 2017. A generalized recurrent neural architecture for text classification with multi-task learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. pages 3385–3391. https://doi.org/10.24963/ijcai.2017/473.

Ye Zhang, Stephen Roller, and Byron C. Wallace. 2016. MGNC-CNN: A simple approach to exploiting multiple word embeddings for sentence classification. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 1522–1527. http://aclweb.org/anthology/N/N16/N16-1178.pdf.