Learning What to Share: Leaky Multi-Task Network for Text Classification

Liqiang Xiao[1,2], **Honglun Zhang**[1,2], **Wenqing Chen**[1,2], **Yongkun Wang**[3], **Yaohui Jin**[1,2]
[1] State Key Lab of Advanced Optical Communication System and Network,
Shanghai Jiao Tong University
[2] Artificial Intelligence Institute, Shanghai Jiao Tong University
[3] Network and Information Center, Shanghai Jiao Tong University
{jinyh}@sjtu.edu.cn

Abstract

Neural network based multi-task learning has achieved great success on many NLP problems, which focuses on sharing knowledge among tasks by linking some layers to enhance the performance. However, most existing approaches suffer from the interference between tasks because they lack of selection mechanism for feature sharing. In this way, the feature spaces of tasks may be easily contaminated by useless features borrowed from others, which will confuse the models for making correct prediction. In this paper, we propose a multi-task convolutional neural network with the Leaky Unit, which has memory and forgetting mechanism to filter the feature flows between tasks. Experiments on five different datasets for text classification validate the benefits of our approach.

1 Introduction

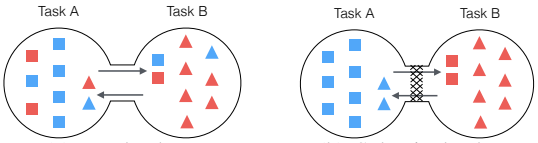
Convolutional neural network (CNN) models have achieved impressive results on many natural language processing (NLP) tasks, which use convolving filters to extract local features and have been successfully applied in computer vision field. In recent years, increasing works transfer CNNs to natural language 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 their numerous parameters. It costs a lot for a large-scale dataset to train 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 insufficiency for annotated resources is not intrinsically solved either.

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; Liu et al., 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).

Proceedings of the 27th International Conference on Computational Linguistics, pages 2055–2065
Santa Fe, New Mexico, USA, August 20-26, 2018.



(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.

The scheme for information sharing is the linchpin for designing a practical multi-task network. Most existing work attempts to find an appropriate proportion to sharing the layers between tasks, despite they entirely reuse some layers among tasks (Liu et al., 2015; Caruana, 1997) or add the tasks' layers up at a proportion (Fang et al., 2017). And recently, the latter architecture shows its advantages for controlling the relation intensity among tasks and becomes prevailing. More models adopt this thought to enhance the performance (Liu et al., 2015; Liu et al., 2016).

However, under the scheme of proportional addition (Ruder et al., 2017; Misra et al., 2016), all the features, between every pair of tasks, are shared in the same weight without selection. Helpless or harmful features if freely transported between tasks with the same importance as helpful ones, namely, the interference is generated just like Figure 1(a) shows. This would burden the network for distinguishing the helpful features and even mislead the predictions.

To resolve above problems, we propose a new CNN-based model *LK-MTL* for multi-task learning, which shares the features with a selective mechanism (Figure.1(b)). Our model employs a "split structure" that every task owns a private subnet and shares their features through a well-designed module—*Leaky Unit*, which has memory and forgetting mechanisms to control the feature flows, deciding what features should be shared or discarded. By that the feature flows are purified and the interference would be restrained.

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

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

- Proposed Leaky Unit has the ability to remember and forget, which is effective for filtering the feature flows and can help multi-task learning dispel the interference among tasks.
- The architecture of our multi-task model combines the advantages of GRU and CNN. The performance is enhanced by both memory mechanism from gated recurrent unit (GRU) and the feature extraction ability from convolutional neural network.
- Our model achieves strong results on several benchmark classification datasets and outperforms the state-of-the-art baselines on four datasets.

2 CNN Models 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; Socher et al., 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). Most of the CNN models for text sequences are

constructed by alternating the convolution layers and pooling layers. In this paper, this kind of CNN architectures is defined as the subnet in our multi-task network, which can be detailed as follows.

2.1 CNN for Text Representation

Given a text sequence $x_{1:l} = x_1 x_2 \cdots x_l$, 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 stacking 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 respectively in fully connected layer. Usually, drop out mechanism is used as a kind of regularization for output layer and the Eq.3 can be rewrote 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 to be 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; $\mathbf{y}$ is the ground-truth label; N and C are the batch size and the number of classes respectively.

3 Multi-Task CNN for Text Classification

The goal of multi-task learning is to utilize the connection among these related tasks to improve classification when learning tasks in parallel. Recently, "split architecture" that provides each task a private subnet is widely accepted by researchers, since it has ability to make a balance between task specific features and shared features. Hence, the key factor of multi-task learning is the fusion mechanism for the information flow between tasks. In CNN models, the information lies in the feature maps. Therefore, the biggest difference of the multi-task architecture is the method for fusing the feature maps. Here we introduce a typical architecture for CNN based multi-task models: PA-MTL.

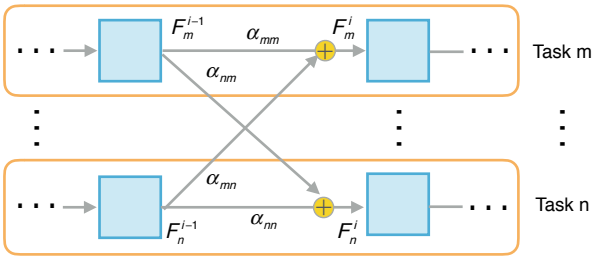


Figure 2: Illustration of proportional addition model.

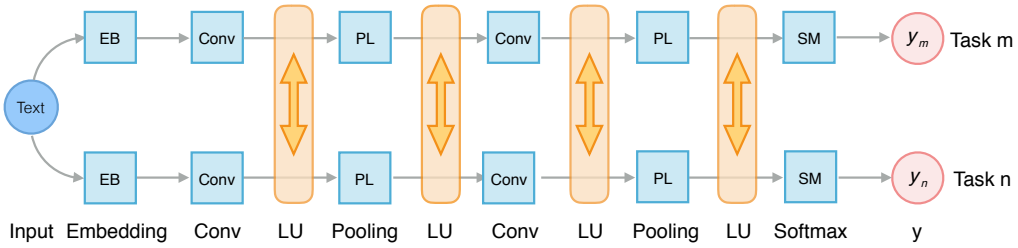


Figure 3: Illustration of the architecture of Leaky Multi-Task Network. LU denotes the Leaky Unit.

Proportional Addition Model (PA-MTL) Generally, most existing multi-task CNN models share the information between tasks by proportional addition (Misra et al., 2016; Fang et al., 2017). As Figure 2 shows, they share the information by adding the feature maps between tasks m, n with scalar weights α^{i-1}. α^{i-1} is updated by back-propagation, which reflects the strength of association between tasks but has no selection for the features. Between the $i-1$-th and i-th layers, the whole process for fusing the feature maps F^{i-1} from M tasks can be formulated as:

$$\begin{bmatrix} \mathbf{F}_1^i \\ \vdots \\ \mathbf{F}_M^i \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \cdots & \alpha_{1M} \\ \vdots & \ddots & \vdots \\ \alpha_{M1} & \cdots & \alpha_{MM} \end{bmatrix} \begin{bmatrix} \mathbf{F}_1^{i-1} \\ \vdots \\ \mathbf{F}_M^{i-1} \end{bmatrix}, \tag{6}$$

4 Leaky Multi-Task CNN

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

4.1 Model Architecture Choice

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 above problem, we explore another structure— *Leaky Multi-Task CNN (LK-MTL)*, in which the tasks have ability to selectively borrow helpful knowledge from others.

As shown in Figure 3, in order to reduce the interference, we assign each task a private subnet, under which circumstance the information is shared indirectly and easy to control. These relatively separated tasks can only borrow information from others through the bridge: *Leaky Unit*. Memory mechanism is employed in this unit to control the feature sharing. The parameters are updated through backpropagation to optimize the selection without needing for extra supervision. This is an end-to-end and easy-training method and its thought is easy to be employed by other works. The convolutional neural network can be easily replaced by multi-layer perceptrons or recurrent neural networks for other applications.

4.2 Leaky Unit

In this separate architecture, we design a module named *Leaky Unit* to selectively share the information between tasks, which is inspirited by the hidden unit of Gated Recurrent Unit (GRU) (Cho et al., 2014), a variant of Leaky Integration Unit proposed by (Bengio et al., 2013). The hidden state of GRU fuses the cell and hidden state of standard LSTM to make the structure easier to compute and implement, which has a sophisticated mechanism for remembering and forgetting features. Thus we make some revision and use it as a tool to filter the feature flows between tasks. Leaky Unit addresses two problems in the process of sharing: what features could be helpfully borrowed from other tasks and How many features should be preserved for current task?

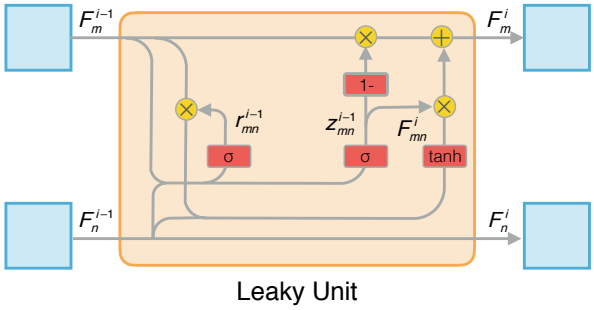


Figure 4: Illustration of the details for Leaky Unit

As shown in Figure 4, a *Leaky Gate* r_{mn} is first generated from the feature maps in $(i-1)$-th layers

$$\mathbf{r}_{mn}^{i-1} = \sigma(\mathbf{W}_r^{i-1} \cdot [\mathbf{F}_m^{i-1}, \mathbf{F}_n^{i-1}]), \tag{7}$$

where mn means the direction of feature flow is from task n to task m; σ denotes the logistic sigmoid function, which limits the values into $[0, 1]$; $\mathbf{F}_m^{i-1}$, $\mathbf{F}_n^{i-1}$ is the feature maps from task m, n respectively. Leaky gate decides what features will be leaked in from other tasks. Hence, a new feature map is calculated by

$$\tilde{\mathbf{F}}_{mn}^i = tanh(\mathbf{U}^{i-1} \cdot \mathbf{F}_m^{i-1} + \mathbf{W}^{i-1} \cdot (\mathbf{r}_{mn}^{i-1} \odot \mathbf{F}_n^{i-1})). \tag{8}$$

When the j-th element in vector $\mathbf{r}_{mn}^{i-1}$ is close to 0, the corresponding feature $[\mathbf{F}_n^{i-1}]_j$ in task n will be discarded. On the contrary, feature $[\mathbf{F}_n^{i-1}]_j$ will be kept and passed to task m.

In addition, we employ another *Update Gate* z_{mn}^{i-1} to determine how much information should be maintained from current task m into the next layer, which is emitted by

$$\mathbf{z}_{mn}^{i-1} = \sigma(\mathbf{W}_z^{i-1} \cdot [\mathbf{F}_m^{i-1}, \mathbf{F}_n^{i-1}]). \tag{9}$$

And final output for current task m is calculated by

$$\mathbf{F}_m^i = \mathbf{z}_{mn}^{i-1} \cdot \mathbf{F}_m^{i-1} + (1 - \mathbf{z}_{mn}^{i-1}) \cdot \tilde{\mathbf{F}}_{mn}^i. \tag{10}$$

If we consider all the directions of information flows and extend above formulations into all M tasks, the output of Leaky Unit is the sum of each row in

$$\begin{bmatrix} \sum_{k=1}^M \mathbf{z}_{1k}^{i-1} & (1-\mathbf{z}_{12}^{i-1}) & \cdots & (1-\mathbf{z}_{1M}^{i-1}) \\ (1-\mathbf{z}_{21}^{i-1}) & \sum_{k=1}^M \mathbf{z}_{2k}^{i-1} & \cdot & 1-\mathbf{z}_{2M}^{i-1}) \\ \vdots & \vdots & \ddots & \vdots \\ (1-\mathbf{z}_{M1}^{i-1}) & (1-\mathbf{z}_{M2}^{i-1}) & \cdots & \sum_{k=1}^M \mathbf{z}_{Mk}^{i-1} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{F}_1^{i-1} & \mathbf{F}_{12}^{i-1} & \cdots & \mathbf{F}_{1M}^{i-1} \\ \tilde{\mathbf{F}}_{21}^i & \mathbf{F}_2^{i-1} & \cdots & \tilde{\mathbf{F}}_{2M}^i \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{F}}_{M1}^i & \tilde{\mathbf{F}}_{M2}^i & \cdots & \tilde{\mathbf{F}}_M^i \end{bmatrix} /M. \tag{11}$$

Leaky units adjust the states of two gates according to the helpfulness of other tasks' features. The units tend to borrow more features from other tasks will more frequently activate the leaky gate. On the contrary, the update gate will be more active to preserve more information in current task.

4.3 Training

In the last layer of model, the vector representations $\hat{\mathbf{F}}_m$ of input sequences are fed into different output layers to fit the number of class, which emits the prediction of probability distribution for task m

$$\hat{y}_m = softmax(\mathbf{W}_m \hat{\mathbf{F}}_m + b_m), \tag{12}$$

where $\hat{y}_m$ 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.

$$\Phi = \sum_{m=1}^{M} \lambda_m L(\hat{y}_m, y_m), \tag{13}$$

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
QC	Question Types	Sentence	5153	-	489	6	10

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

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.

5 Experiments

In this section, we investigate the empirical performances of our models on five related benchmark tasks for text classification. And the results are compared with the state-of-the-art models. Furthermore, we also intuitively show the operation mechanism of leaky unit via visualization.

5.1 Datasets

As Table 1 shows, we collect five benchmark datasets for text classification that are related to each other to different degree. These datasets contain sentiment, subjectiveness 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 neutral reviews removed and 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 document-level movie reviews that are classified to be positive/negative.

- **QC** Question dataset classifying a given question into six types (whether the question subjects to person, location, numeric information, etc.) (Li and Roth, 2002).

[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/

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

Table 2: Hyperparameter settings

Model		SST-1	SST-2	IMDB	SUBJ	QC
Single-Task	RNTN	45.7	85.4	-	-	-
	DCNN	48.5	86.8	-	-	-
	MGNC-CNN	48.7	88.3	-	94.1	-
	PV	44.6	82.7	**91.7**	90.5	91.8
Multi-Task[4]	MT-GRNN	49.2	87.7	91.6	-	92.3
	MT-RNN	49.6	87.9	91.3	94.1	-
	MT-CNN	49.0	86.9	91.5	94.1	92.0
	MT-DNN	48.1	87.3	91.4	94.0	92.1
	PA-MTL	48.2	87.1	90.7	94.0	91.5
Our Models	LK-MTL (hidden unit)	49.4	**88.6**	91.1	94.3	92.9
	LK-MTL (hidden unit + peephole)	49.6	88.2	91.3	94.1	**93.8**
	LK-MTL (leaky unit)	**49.7**	88.5	91.3	**94.5**	**93.8**

Table 3: Results of LK-MTL against other state-of-the-art models.

5.2 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., 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. The key hyperparameters are chosen via a small grid search, and final setting is illustrated in Table 2.

5.3 Performance of Multi-task CNN

We simultaneously train our model LK-MTL on five datasets and compare it with single-task scenario in Table 3. We can see that our model improves the performance with a large margin over the single task models, which demonstrates the positive synergy of our multi-task architecture. This also testified our assumption that separate structure for MTL is conducive to the information sharing between tasks, helping making better representations for text sequences.

We also test several variants of LK-MTL. The first one replaces Leaky Unit by the Hidden Unit [5] of standard LSTM, and the second one further adds the peephole connections. The results reported in Table

[4]We use reported data for MT-GRNN and MT-RNN and implement MT-CNN, MT-DNN and PA-MTL since the authors do not release the code.

[5]To suit the multi-task learning architecture, hidden state h_t is removed from the formulations of hidden unit.

3 shows that our LK-MTL with Leaky Unit outperforms its variants, which proves the mechanism of Leaky Unit is more suitable for multi-task CNNs.

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 3, LK-MTL also outperforms the state-of-the-art multi-task learning 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. Specifically, LK-MTL surpasses the PA-MTL, which result demonstrates that the memory and forgetting mechanisms in leaky unit indeed have advantages over other sharing approaches, which help distinguish the helpfulness of the features and making a balance between features in current task and borrowed features. In another word, task-specific feature and shared feature is organically fused by leaky unit in a better way.

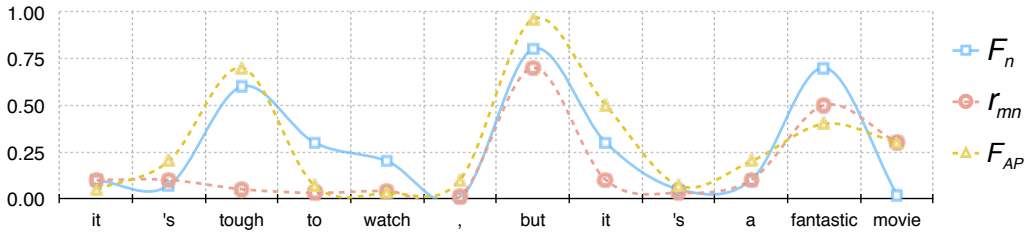


Figure 5: Lines illustrate the feature weights of $\mathbf{F}^1_{SUBJ}$ in SUBJ subnet. And red line shows the value of $\mathbf{r}^1_{SST-1 \leftarrow SUBJ}$ that filters the features from SUBJ subnet to SST-1 subnet. F_{AP} line visualizes the feature weights in the first layer of PA-MTL.

5.4 Visualization

To intuitively show the selection process in leaky unit, we design an experiment to show the values of gate and how they block the useless features. For the first convolutional layer and leaky unit, we visualize the activations $\mathbf{F}^1_m$ of the filters with normalized values and show their corresponding leaky gate $\mathbf{r}^1_{mn}$ in the gate units. By that we can easily find what kind of features are discarded as interference.

Figure 5 illustrates the behavior of leaky unit on a random selected sentence from test set of SST-2 task. We visualize the results of the first feature map for SUBJ subnet and the leak gate $\mathbf{r}^1_{mn}$ that filters the features from SUBJ to SST-2 task. For the positive sentence "it's tough to watch, but it's a fantastic movie", we can see that subnet for SUBJ task focuses on two critical positions "tough" and "fantastic". The word "tough" is subjective for SUBJ task, so the subnet focuses on that world, but actually it is useless and may mislead the prediction of SST-2 task. Successfully, our leaky gate lowers the intensity of that interference "tough", making a correct prediction. However, PA-MTL wrongly makes a negative prediction for lacking resistance to interference. This indicates the effectiveness of our memory and forgetting mechanism in the leaky unit.

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; Liu et al., 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; Fang et al., 2017) proposes a model for images, trying to reduce the interference through weighting the connections between tasks and finding an appropriate proportion. Though this kind of methods weakens the noise from less-related tasks, the processing is coarse-grained and cannot distinguish the useful information in smaller feature level.

Different from these models, our model control the information flows between tasks in feature level, which uses the mechanism of remembering and forgetting, passing only the useful features for the current task. It solves the problems of what should be maintained in current task and what should be borrowed from other tasks. By that our model is capable of reducing the interference without extra supervision.

7 Conclusion and Future Work

In this paper, we propose a CNN based framework for multi-task learning, which has the structure to control the passing of information in feature level. By remembering the helpful features and forgetting the useless ones, LK-MTL can reduce the interference between the tasks. And we also visualize the property of our models and intuitively show the selection mechanism of Leaky Unit. Experiment result on five datasets for text classification demonstrates the effectiveness of our model for text representation.

In the feature, we would like to investigate deeper into the interference problem, testing more kinds of gate mechanisms and find better one to purify the feature flows. We also want to equip our unit on other neural networks for other applications, such as multi-layer perceptrons and RNNs.

8 Acknowledge

We appreciate the constructive advices from Naoki Yoshinaga and the valuable comments from anonymous reviewers. We also thank Xuan Luo for building and maintaining the GPU platforms. This research was funded by National Natural Science Foundation of China under Grant No. 61371048.

References

Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. 2013. Advances in optimizing recurrent networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8624–8628.

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

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Computer Science*.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, 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.

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.

Yuchun Fang, Zhengyan Ma, Zhaoxiang Zhang, Xu-Yao Zhang, and Xiang Bai. 2017. Dynamic multi-task learning with convolutional neural network. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1668–1674.

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.

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.

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.

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.

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

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)*, June.

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.

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.

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.

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.

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

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.

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.