Feature Projection for Improved Text Classification

Qi Qin[1,2,*], **Wenpeng Hu**[3,*], **Bing Liu**[2,†]
[1] Center for Data Science, AAIS, Peking University
[2] Wangxuan Institute of Computer Technology, Peking University
[3] Department of Information Science, Peking University
{qinqi, wenpeng.hu, dcsliub}@pku.edu.cn

Abstract

In classification, there are usually some good features that are indicative of class labels. For example, in sentiment classification, words like *good* and *nice* are indicative of the positive sentiment and words like *bad* and *terrible* are indicative of the negative sentiment. However, there are also many *common features* (*e.g.*, words) that are not indicative of any specific class (*e.g.*, voice and screen, which are common to both sentiment classes and are not discriminative for classification). Although deep learning has made significant progresses in generating discriminative features through its powerful representation learning, we believe there is still room for improvement. In this paper, we propose a novel angle to further improve this representation learning, i.e., *feature projection*. This method projects existing features into the orthogonal space of the common features. The resulting projection is thus perpendicular to the common features and more discriminative for classification. We apply this new method to improve CNN, RNN, Transformer, and Bert based text classification and obtain markedly better results.

1 Introduction

Text classification is an important task in natural language processing and text mining. It has a very wide range of applications, such as sentiment classification (Liu, 2012), question classification (Li and Roth, 2002), and deception detection (Liu, 2012; Feng et al., 2012). In recent years, deep learning models have been shown to outperform traditional classification methods (Kim, 2014; Iyyer et al., 2015; Tang et al., 2015; Dai and Le, 2015; Jin et al., 2016; Joulin et al., 2017; Shen et al., 2018). Given the input document, the system applies a mapping function (*e.g.*, averaging or summation, a convolution neural network (CNN), recurrent neural network (RNN), and so on) to learn a dense representation of the document and then uses this representation to perform the final classification. Representation learning is one of the key strengthes of deep learning.

In this paper, we propose to further improve the representation learning, i.e., to make the representation more discriminative for classification. Note that throughout the paper we will use sentence sentiment classification as an example to explain different ideas, but in our experiments, non-sentiment classification datasets are also used to show the generality of the proposed method. For text classification, many neural networks and embedding techniques have been devised and applied, e.g., RNN, CNN, Transformer (Vaswani et al., 2017) and Bert (Devlin et al., 2018). For example, RNN can model the whole sentence and also capture the long-term dependencies within the sentence. However, modeling the entire sequence may neglect some key local contexts that are important for classification (Yin et al., 2017). CNN is able to extract more local and position-invariant features (Scherer et al., 2010; Collobert et al., 2011). However, these methods may not give enough weights to some special or discriminative words. To solve this problem, the attention mechanism was introduced. For example, by exploiting attention, Transformer and Bert (which maximizes Transformer's ability to extract sentence semantic information) can achieve even better results than both CNN and RNN on many tasks. We will see some other related methods to produce effective representations in the related work section.

Although the existing models are already able to produce excellent representations, we will show that these representations can still be improved. This paper explores in an entirely different direction, i.e., feature projection. In a typical sentence or

[*] Equal Contribution.
[†] Corresponding Author.

document, there are usually some words or features that are correlated with some class labels, but there are also many other common features that cannot distinguish different classes. For example, in sentiment classification, words like *Good* and *Nice* are indicative of the positive sentiment, and words like *Bad* and *Terrible* are indicative of the negative sentiment. Words like *picture*, *price*, and *battery* are not indicative of any sentiment, i.e., they are not discriminative. However, they may still interfere the representation learning to produce sub-optimal feature representations for the final classification. Even though the attention mechanism can alleviate this problem to some extent by giving higher weights to words associated with classes and lower weights to the other words that are not indicative of any specific classes. However, due to the idiosyncrasy of the data and the inaccuracy of the attention mechanism, the problem remains.

In this paper, we propose a novel feature projection method to improve feature representation learning to make it more discriminative for classification. The proposed method is called Feature Purification Network (**FP-Net**). Specifically, FP-Net consists of two sub-networks, a *common feature learning network* referred to as the C-net and a *projection network* referred to as the P-net. C-net uses a Gradient Reverse Layer (**GRL**) (Ganin and Lempitsky, 2014; Zhang et al., 2019) to extract common features $\vec{b}$ (*i.e.*, invariant features (Zhang et al., 2019)) that are shared by multiple classes and have little discriminative power for classification. At the same time, P-net uses a traditional feature extractor to learn the feature vector $\vec{a}$ for the input sentence or document. Then the feature (or representation) vector $\vec{a}$ is projected onto the vector of the common features $\vec{b}$ (*i.e.*, vector $\vec{b}$) to get a projection vector $\vec{c}$, which represents the input sentence's own common features. Then, we project the feature vector $\vec{a}$ onto the orthogonal direction of the vector of the common features $\vec{c}$ to produce the final purer features for classification. It is quite clear and intuitive that this orthogonal project is to get rid of the common features and make the system focusing on those discriminative features only. We will explain why two projections are used in Section 3.

In summary, the key contribution of this paper is the improvement to representation learning through feature vector projection. To the best of our knowledge, this is the first such technique. Specifically, an Orthogonal Projection Layer (**OPL**) is proposed to map the features obtained by a traditional feature extractor to the classification-specific semantic space, which is orthogonal to the common features such that we obtain a more relevant and discriminative (or purer) feature representation from the original document for classification.

Extensive experiments have been conducted to verify the effectiveness of the proposed method on two sentence sentiment classification datasets MR and SST2, a natural language inference dataset SNLI, and a question classification dataset TREC. The results show that the proposed method can improve the classification accuracy of RNN, CNN, Transformer and Bert based classification methods markedly, which shows that feature projection is a highly promising direction to explore.

2 Related Work

It is well known that one of the key strengths of deep neural networks is their superb ability to learn highly effective representations or features from the raw data, which have been shown to be very successful for all kinds of applications including natural language processing tasks such as text classification (Jin et al., 2016), machine translation (Bahdanau et al., 2014; Vaswani et al., 2017) dialogue (Wang and Jiang, 2016), etc. Previous work on learning representations broadly falls in two main categories: supervised and unsupervised methods. Our work focuses on improving the representation of text for supervised classification.

Supervised methods: These methods improve data utilization efficiency and discriminative feature distillation as they can obtain better training signals from the labeled data. Sequence models such as recurrent neural networks (RNN), Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Chung et al., 2014) networks are suitable for handling text because a sentence or document can be regarded as a sequence. Therefore, a large amount of work based on RNN and its variants for feature extraction and downstream tasks has been done (Tang et al., 2015; Wang and Tian, 2016; He et al., 2016). Unlike RNN's sequence modeling approach, CNN (Convolutional Neural Network) uses different sized windows to capture local correlations and position-invariant information (Kim, 2014; Conneau et al., 2016; Lai et al., 2015; Xiao and Cho, 2016; Wang, 2018). A common approach of these methods is to create an instance-level representation by using the final hidden state of the RNN, the maximum (or average) pooling of the RNN hidden states, or convolutional n-grams. However, they may ignore the importance of special words that are highly discriminative for classification. After Bahdanau et al. (2014) introduced the attention mechanism in machine translation, attention mechanism has been exploited in many natural language processing tasks including text classification to solve the above problem. For example, Yang et al. (2016) introduced attention as an integral part of the model for text classification. Lin et al. (2017) proposed a new model for extracting interpretable sentence embeddings using self-attention. Ma et al. (2018) showed that attention mechanism is also effective for sentiment classification. Vaswani et al. (2017) further illustrated that they can get a stronger sentence-level representation by stacking multiple blocks of self-attention. Bert (Devlin et al., 2018) combines Transformer and a large corpus to produce an even more complete and better sentence-level representation. Some other studies improved the representation of sentences from the perspective of language structures (*e.g.*, parse trees and dependency trees) (Tai et al., 2015; Mou et al., 2015). Subramanian et al. (2018) utilized a single multi-task framework to combine the benefits of diverse sentence representation learning objectives. However, to the best of our knowledge, these existing works and others have not used feature projection to improve (or purify) representations for supervised learning, which we believe is a promising direction to explore.

Unsupervised methods: These methods utilize a large unlabeled text corpus to learn word representations which are then composed into sentence and document representations. For example, Kiros et al. (2015) constructed sentence representations by trying to reconstruct neighbouring sentences. Hill et al. (2016) proposed a log-linear bag-of-words models for sentence representation. The unsupervised smooth inverse frequency method in (Ethayarajh, 2018) built on this but used a weighted average of word embeddings and principal component removal for sentence representations. Our work is again clearly different from these unsupervised methods as the proposed method works under supervised learning. Existing unsupervised methods also do not use feature projection.

Some other works have also been done for semi-supervised representation learning (Kevin Clark, 2018) and transfer learning (Tamaazousti et al., 2018). Jason Phang (2019) also proposed to use some data-rich intermediate supervised tasks for pre-training to help produce better representation for the end task. To the best of our knowledge, all these previous studies tried to improve representations using external data or knowledge, which are quite different from our method as we don't use any external information. Also, the philosophy of our approach is entirely different as we try to eliminate commonalities among classes through feature projection, which is orthogonal to existing representation learning approaches.

Finally, our work is related to several other works. Ganin and Lempitsky (2014) introduced the gradient reverse layer (GRL) for extracting common features in the context of domain adaptation. It embeds domain adaptation into the process of learning representations so that the final classification decision has more discriminative and invariant characteristics for domain changes. We also use GRL to extract irrelevant or common features. However, we do not work on domain adaptation and they do not use feature projection. Belinkov et al. (2019) used adversarial learning to encourage models to learn representations free of hypothesis-only biases in the SNLI dataset. Zhang et al. (2019) combined GRL and aspect attention to study cross-domain sentiment classification. They found common features across domains and then extracted information from the aspects (which are product features) with the help of common features to do classifications. Our work is clearly different because none of these existing works improve representation learning through feature projection.

3 Feature Purification Network

The overall framework of our model is shown in Figure 1. The whole model consists of two parts, the first part is the *projection network* (*i.e.*, P-net) and the other is the *common feature learning network* (*i.e.*, C-net). As mentioned earlier, the goal of C-net is to extract common features and the goal of P-net is to compute the purified features for classification, which is done by projecting the learned full information vector of the input document into a more discriminative semantic space to eliminate the influence of the common features.

P-net consists of four parts: the input layer X, the feature extractor F_p, Orthogonal Projection

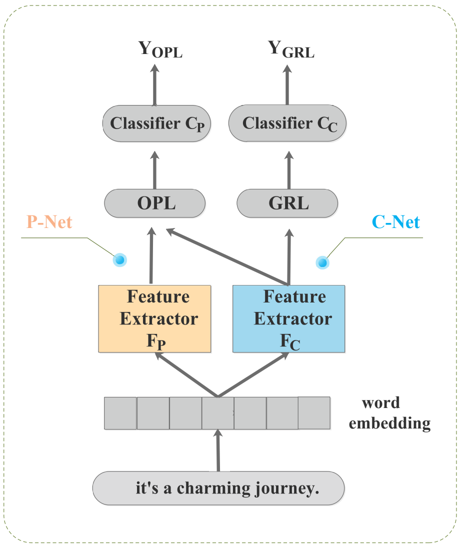

Figure 1: The architecture of FP-Net

Layer (**OPL**), and the final classification layer C_p. C-net is also composed of four parts: the input layer X, the feature extractor F_c (F_p and F_c's parameters are not shared)[1], the Gradient Reverse Layer (**GRL**) and the classification layer C_c. The key idea of the proposed technique is as follows: The feature vector f_p computed by the feature extractor F_p is projected to the orthogonal direction of the feature vector f_c extracted by F_c of the C-net. That is, f_p (the full information extracted from the input document) is projected to the discriminative semantic space to be purified for the final classification. However, in order to perform the orthogonal projection, two operations are required, which we will explain shortly. Next we use CNN as an example feature extractor to detail each component of the proposed FP-Net.

CNN Extractor: Given a dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i is an input document with the length L (after padding or cut) and y_i is the label corresponding to the sample x_i. Let $V_{ij} \in \mathbf{R}^k$ be the word vector corresponding to the jth word of the document x_i. $X_i \in \mathbf{R}^{L\times k}$ is the embedding matrix of x_i. Recall our FP-Net model consists of two sub-networks, i.e., P-net and C-net, with the same input x_i. The two sub-networks also have the same structure for the feature extractor CNN, but there are no shared parameters between them. The feature extractors of P-net and C-net are F_p, F_c.

[1]The feature extractor can be any existing extractor. In this work, we verified the effectiveness of our purification network using CNN, RNN, Transformer, and Bert as feature extractors as we will see in the experiment section.

We use F_c as an example to introduce the working of CNN. When the feature extractor F_c receives X_i from the input layer, F_c extracts the advanced features f_c from X_i in the form of n-grams, which is:

$$f_c = [c_1, c_2, ..., c_{l-n+1}] = [c_j]_{j=1}^{l-n+1}, \qquad (1)$$

where c_j represents the output produced by CNN's filter on $X_i[j : j+n-1, :]$. Mathematically, a convolution operation consists of a filter $W \in \mathbf{R}^{n\times k}$ and a bias $b \in \mathbf{R}$. Then c_j can be expressed as:

$$c_j = g(W \cdot X_i[j : j+n-1, :] + b), \qquad (2)$$

where g is a nonlinear activation function such as **Relu**. We use a Maxpooling operation over the feature map and take the maximum value $f_c = max\{f_c\}$ as the feature corresponding to this particular filter. The same feature extractor F_p will also get the advanced features f_p from the input layer. We refer to the features of the P-net and C-net respectively as

$$f_p = \mathrm{CNN_P}(X), \qquad (3)$$

$$f_c = \mathrm{CNN_C}(X). \qquad (4)$$

Other details of C-net will be introduced in **C-net Module**, and likewise, additional details about P-net will be introduced in **P-net Module**.

C-net Module: The goal of C-net is to extract the common features, which are the semantic information of the input example that is not discriminative for the classification task. As mentioned earlier, common features are those shared by all classes of the problem. The classifier C_c should not use them to distinguish different classes. To obtain common features, we add a Gradient Reverse Layer (**GRL**) (Ganin and Lempitsky, 2014; Ganin et al., 2016) after the feature extractor F_c to reverse the gradient direction. Through this training module, we can obtain the common features that are shared among classes.

Without loss of generality, we can think of the gradient reverse layer as a "pseudo-function" defined by two incompatible equations describing its forward and back-propagation behaviors:

$$GRL(x) = x, \qquad (5)$$

$$\frac{\partial GRL(x)}{\partial x} = -\lambda I, \qquad (6)$$

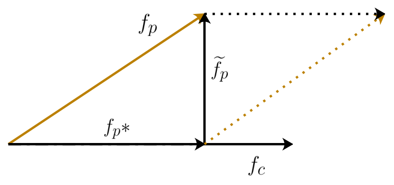

Figure 2: Working of the Orthogonal Projection Layer. The example here is in a 2-dimensional space. f_p represents the traditional feature vector; f_c represents the common feature vector; f_{p*} is the projected feature vector; $\widetilde{f_p}$ is our final Orthogonal Projection feature vector.

where λ is a hyper-parameter. We process the feature vector f_c through **GRL** as $GRL(f_c) = \widetilde{f_c}$, which is then fed to the classifier C_c:

$$Y_{GRL} = \text{softmax}(\widetilde{f_c} \cdot W_c + b_c), \tag{7}$$

$$Loss_c = \text{CrossEntropy}(Y_{truth}, Y_{GRL}), \tag{8}$$

where W_c and b_c are the weights and bias of C_c respectively. By optimizing the objective function $Loss_c$, the feature extractor F_c is able to extract the common features of different classes.

P-net Module: The goal of P-net is to first extract the full semantic information from the input example and then project it into the semantic space purified for classification. In order to achieve this, we perform the projection of the feature f_p extracted by the feature extractor F_p onto the orthogonal direction of the common feature f_c, extracted by F_c. The feature space orthogonal to the common feature vector should contain features that are pure and highly effective for classification (*e.g.*, sentiment related information in sentiment classification). Projecting the traditional feature vector f_p to this orthogonal feature space preserves the discriminative information and remove those common features of the classes that are unhelpful and even confusing to the classification task.

The Orthogonal Projection Layer (**OPL**) helps us accomplish this goal. Figure 2 illustrates the idea of OPL using a two-dimensional space example. Mathematically, we first project the tradition feature vector f_p onto the common feature vector f_c:

$$f_p* = \text{Proj}(f_p, f_c), \tag{9}$$

where Proj is a projection function.

$$\text{Proj}(x, y) = \frac{x \cdot y}{|y|} \frac{y}{|y|}, \tag{10}$$

where x, y are vectors. We then do the projection in the orthogonal direction of the projected feature f_p to get the purer classification feature vector:

$$\widetilde{f_p} = \text{Proj}(f_p, (f_p - f_p*)). \tag{11}$$

Clearly, it is easy to show that the feature vector $\widetilde{f_p}$ obtained by Eq. 11 is equivalent to f_p–f_p*. Using the traditional feature vector f_p and the projected feature vector f_p*, we can build a plane (in **three dimensions**). The intersection of this plane and the orthogonal plane of the projected feature vector f_p* is our pure feature vector. In other words, the projection in Eq. 9 is a constraint on the common feature vector. That is to say: the modulus of the common feature vector is limited by projecting the traditional feature vector of the input x_i to the common feature vector, so the semantic information of the new common feature vector (*i.e.*, the projected feature f_p*) contains only the common semantic information in x_i. This makes the final purified feature vector $\widetilde{f_p}$ coming from the traditional feature vector f_p rather than any vector in any plane orthogonal to the common feature vector f_c. Finally, we use the purified feature vector $\widetilde{f_p}$ to do the classification.

$$Y_{OPL} = \text{softmax}(\widetilde{f_p} \cdot W_p + b_p), \tag{12}$$

$$Loss_p = \text{CrossEntropy}(Y_{truth}, Y_{OPL}). \tag{13}$$

Note that here $Loss_p$ and $Loss_c$ are trained simultaneously, and they use different optimizers. $Loss_p$ uses the Adam optimizer. Since Ganin and Lempitsky (2014) used Moment SGD as the domain classifier's optimizer, our C-net loss function $Loss_c$ also uses Moment SGD optimizer.[2] Gradients are also passed back through feature f_c when optimizing $Loss_p$. Although the two losses are opposite to each other in terms of optimization targets of the feature extractor F_c, the effect of $Loss_p$ on F_c is in the orthogonal direction of f_c. A balance will be found to make the extracted feature f_c closer to the real common features. The complete training algorithm of the proposed FP-Net is given in Algorithm 1, which is self-explanatory.

4 Experiments

We now evaluate the proposed FP-Net [3] using four text classification datasets and compare it with baselines without the purification capability. Our goal is

[2]We have conducted experiments using the Adam optimizer for both C-Net and P-Net. The results are about the same as using two different optimiers.

[3]`https://github.com/Qqinmaster/FP-Net/`

Algorithm 1 Feature Purification Network

```
1:  Input:
      Dataset $D = \{(x_i, y_i)\}_{i=1}^N$, $x_i$'s embedding matrix $X_i \in \mathbf{R}^{Lk}$; Randomly initialized FP-Net's parameters $\theta$.
2:  for each iteration $b = 1, 2, ..., M$ do
3:      Sample one batch $X_b$ from $D$
4:      C-net part:
5:      Generate common features (CFs) (Eq. 3)
6:      CFs go through GRL (Eq. 5)
7:      Perform classification (Eq. 7)
8:      P-net part:
9:      Generate traditional features (TFs) (Eq. 4)
10:     TFs projection (Eq. 9)
11:     Get the purified features (Eq. 11)
12:     Perform classification (Eq. 12)
13:     Update parameters:
14:     C-net, P-net's parameters are updated together (Eq. 8 & Eq. 13)
15: end for
```

to verify whether the proposed feature purification is general and effective for different deep learning classification models (or more precisely, feature extractors) on diverse datasets.

4.1 Experimental Datasets

We carried out experiments on four diverse benchmark datasets:

MR: This is a movie review dataset for sentiment classification. It has two classes: positive and negative (Pang and Lee, 2005).[4]

SST2: This is the Stanford Sentiment Treebank dataset.[5] Each sample is marked as negative or positive.

TREC: This is a question classification dataset, which is to classify a question into one of the six question types (Li and Roth, 2002).[6]

SNLI: This is a popular text entailment dataset. It contains 570k human annotated sentence pairs, in which the premises are drawn from the captions of the Flickr 30 corpus and hypotheses are manually annotated (Bowman et al., 2015). For this SNLI dataset, we created the following settings to suit our needs: (1) we concatenated the two sentences (in a pair) as a single sample; (2) when using Bert as a feature extractor, we reduced the number of training set samples to 25,000 to speed up the training process. For other feature extractors (see below), the complete data is used.

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

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

[6]http://cogcomp.cs.illinois.edu/Data/QA/QC/

Data	c	l	$Train$	$Test$	$\|V\|$
MR	2	45	8,529	1,066	17,884
SNLI	3	40	54,936	9,824	33,944
SST2	2	35	6,920	1,821	16,789
TREC	6	15	5,000	952	8,834

Table 1: Dataset statistics. c: number of classes. l: average length of sentences, after padding and cutting. $Train$, $Test$: number of training and testing examples. $|V|$: vocabulary size.

The dataset statistics are given in Table 1.

4.2 Baselines

Since our goal is to perform feature purification so that the purified features are more conducive for classification, to verify the validity of the proposed FP-Net model, we compare the classification results with and without purification using the following popular feature extractors:

LSTM: The long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997) for solving the gradient disappearing problem of the traditional RNN.

CNN: We use the Convolution Neural Networks in (Kim, 2014) as the feature extractor to generate representations.

Transformer: We use the encoder part of the model proposed by (Vaswani et al., 2017) as the feature extractor, followed by a classifier.

Bert: We fine-tuned on the trained Bert base (Devlin et al., 2018). Bert base includes 12-layer, 768-hidden, 12-heads and 110M parameters. In particular, we use Bert-base Uncased, where Uncased means that the text has been lower cased before WordPiece tokenization.

Note, those existing feature learning or feature enhancement approaches discussed in Section 2 are not compared as they are entirely different from our approach. They mainly relied on external data or information to improve representation learning. Our method does not use any external data or information. However, we do include Bert as a baseline as it is perhaps one of the most successful feature learning methods using external data. Our method can improve on top of Bert.

Model	MR	SNLI	SST2	TREC
LSTM	77.46(±0.41)	76.98(±0.07)	80.41(±0.20)	87.19(±0.58)
FP+LSTM	78.13(±0.18)	77.92(±0.10)	81.60(±0.17)	88.83(±0.40)
CNN	76.18(±0.45)	72.92(±0.19)	80.47(±0.59)	90.86(±0.51)
FP+CNN	78.74(±0.36)	74.38(±0.14)	82.02(±0.11)	92.78(±0.26)
Trans	75.18(±0.57)	66.71(±0.58)	76.93(±0.39)	87.33(±0.23)
FP+Trans	76.83(±0.66)	73.34(±0.43)	78.42(±0.49)	89.51(±0.79)
Bert	87.45(±0.51)	80.78(±0.42)	90.38(±0.10)	96.67(±0.22)
FP+Bert	90.56(±0.35)	81.47(±0.26)	92.24(±0.29)	98.33(±0.24)

Table 2: Results of our FP-Net against baseline methods. In each block, FP+X is a model obtained by our FP-Net using X as the feature extractor. Accuracy (%) is the evaluation metric. Each result in the table is the average accuracy of five experiments with the standard deviation in parentheses.

4.3 Implementation Details

First, all the word embeddings in our experiments are randomly initialized as 200-dimension vectors and then modified during training (except Bert). For each type of feature extractor, we have the following configuration:

1) For the RNN-based models, we use a two-layer LSTM for feature extraction and the hidden state of each layer is set to 256.

2) For the CNN-based models, in order to obtain more fine-grained features, we use filter sizes of [2,3,4,5,6] with 100 feature maps each.

3) For the Transformer-based models, we use Transformer's encoder as the feature extractor, specifically with single-head and 3 blocks.

4) For the Bert-based models, we fine-tuned the pre-trained Bert-base parameters. These settings are exactly the same in the baseline as in our FP-Net.

In the training of the C-net module, we use a stochastic gradient with 0.9 as the momentum and the following annealing learning rate (Ganin and Lempitsky, 2014).

$$l_p = \frac{l_0}{(1 + \alpha \cdot p)^{\beta}}$$

where p is the training progress linearly changing from 0 to 1, $l_0 = 0.01$, $\alpha = 10$ and $\beta = 0.75$. In GRL, the hyper-parameters λ swept $[0.05, 0.1, 0.2, 0.4, 0.8, 1.0]$.

4.4 Experiment Results

In our experiments, we adopt the classification accuracy as the evaluation metric. We summarize the experimental results in Table 2, where FP+X means that the model trained by the proposed FP-Net using X as the feature extractor. Each of the two lines compares the experimental results of the traditional model with our proposed model on these four datasets. From Table 2, we can make the following observations.

1. Our FP-Net model consistently improves the results of the baseline feature extractors (*i.e.*, LSTM, CNN, Transformer and Bert) using the proposed feature projection. This verifies the effectiveness of the proposed feature purification method of projecting the traditional feature vectors to the orthogonal direction of the common features.

2. Compared with the traditional CNN, the FP+CNN model increases the accuracy by 2.56% on the MR dataset and 1.46% on the SNLI dataset. The improvement of FP+LSTM is less, increased by 0.67% and 0.94% on the MR and SNLI datasets. This shows that the way that CNN extracts input features (concatenate the feature after using different sliding window sizes for extracting local features) is quite effective in extracting more complete semantic information, which leads to more irrelevant features being used. That is why the projection on the CNN features brings more improvements compared to the RNN-based model.

3. By comparing the experimental results of the attention-base model (*i.e.*, Transformer and Bert), we can see that our FP-Net can improve the feature representation capabilities of these feature extractors. For example, in the Bert-based experiment, our FP+Bert can increases the accuracy by 3.11% on MR and 1.66% on TREC. That is to say our orthogonal projection method can make the representation of attention-based obtain a higher discriminative power for classification. Outperforming Bert is particularly significant because Bert is perhaps one of the best feature extractors, if not the best.

Model	MR	SNLI	SST2	TREC
FP+CNN	$78.74(\pm0.36)$	$74.38(\pm0.14)$	$82.02(\pm0.11)$	$92.78(\pm0.26)$
FP+CNN-G	$77.71(\pm0.44)$	$72.85(\pm0.62)$	$81.09(\pm0.17)$	$91.89(\pm0.10)$
FP+CNN-O	$76.64(\pm0.39)$	$73.11(\pm0.22)$	$81.25(\pm0.11)$	$90.76(\pm0.37)$
FP+CNN-G-O(plus)	$76.38(\pm0.45)$	$73.08(\pm0.19)$	$80.67(\pm0.52)$	$90.89(\pm0.41)$
FP+CNN-G-O(concat)	$76.18(\pm0.51)$	$72.91(\pm0.26)$	$81.02(\pm0.18)$	$91.16(\pm0.41)$

Table 3: Ablation experiments. The first block contains the results of FP+CNN with GRL (-G) or OPL (-O) removed and the results with both GRL and OPL (-G-O) removed and the features of the two modules (or sub-networks) summed. The second block contains the results with both GRL (-G) and OPL (-O) removed and the features of the two modules concatenated.

Model	MR	SNLI	SST2	TREC
CNN	$76.18(\pm0.45)$	$72.92(\pm0.19)$	$80.47(\pm0.59)$	$90.86(\pm0.51)$
CNN_Dp	$76.72(\pm0.50)$	$73.49(\pm0.14)$	$80.67(\pm0.40)$	$90.91(\pm0.41)$
FP+CNN	$78.74(\pm0.36)$	$74.38(\pm0.14)$	$82.02(\pm0.11)$	$92.78(\pm0.26)$
Trans	$75.18(\pm0.57)$	$66.71(\pm0.58)$	$76.93(\pm0.39)$	$87.33(\pm0.23)$
Trans_Dp	$75.75(\pm0.31)$	$68.36(\pm0.25)$	$77.10(\pm0.48)$	$88.16(\pm0.34)$
FP+Trans	$76.83(\pm0.66)$	$73.34(\pm0.43)$	$78.42(\pm0.49)$	$89.51(\pm0.79)$

Table 4: Experimental results with doubled parameter size on the four datasets. For example, Trans_Dp shows the increase of the number of blocks of the Transformer from 3 to 6.

4.5 Ablation Experiments and Analysis

In order to analyze the effectiveness of each component of FP-Net, we performed the following two ablation experiments.

First, in Table 3, we report the results of the ablation test of each component of FP-Net, where FP+CNN-G (or O, G-O) represents FP-Net with the GRL (or OPL, or both GRL and OPL) removed while using CNN as the feature extractor. The parameters of all the experiments compared in the first block are exactly the same. In order to keep the parameter size consistent, we performed element-wise summation of the features of FP-Net's two sub-networks f_p and f_c in the FP+CNN-G-O experiment. By comparing the experimental results of the first block, we observe the following:

1) Whether GRL or OPL is removed or both GRL and OPL are removed at the same time, the accuracy will drop significantly compared with the complete FP-Net. For example, for the MR dataset, when we remove the GRL and keep the OPL (*i.e.*, FP+CNN-G), the accuracy decreases by 1.03%; When we remove both GRL and OPL, and then execute $f_p + f_c$ (*i.e.*, FP+CNN-G-O(plus)), the accuracy decreases by 2.36%, etc. These results show that each component in FP-Net is important, and the absence of any one component will lead to decline in accuracy.

2) In the experiment of FP+CNN-O, we remove OPL and keep GRL, which means that we use $f_p - f_c$ instead of the orthogonal projection (*i.e.*, $f_p - f_{p*}$). As stated in **P-Net module** of Section 3, such a replacement will give up a constraint that gets the common feature f_{p*} of the current input x_i from the base common feature f_c. The results showed that the accuracy decreases by 2.10% on MR and decreases by 1.27% on SNLI, which mean that the projection operation (*i.e.*, Eq. 9) is necessary.

3) Clearly, adding f_p and f_c of FP-Net is not the only way to connect the two sub-networks of FP+CNN-G-O. We can do $f_p \oplus f_c$, where $\oplus$ is the concatenation operator. Although this method has more parameters in the P-net classifier, we can still observe that the accuracy of FP+CNN-G-O is not as good as the accuracy of FP+CNN. For example, FP+CNN-G-O reduced the accuracy by 2.36% on MR and 1.30% on SNLI, which can also prove the effectiveness of GRL and OPL in our FP-Net.

Second, we show that the improvement in accuracy by FP-Net is not due to the increase in the number of parameters. We doubled the parameters of traditional CNN and Transformer and compared with our FP+CNN, FP+Trans. The results of this part of the experiments are shown in Table 4, where the index 'Dp' means the Doubled parameter size. For example, Tans_Dp increases the number of blocks of Transformer in the baseline from 3 to 6. All experimental results show that increasing the number of parameters of the baseline models will improve classification accuracy slightly, but there is still a large gap with the proposed model.

5 Conclusion

In this paper, we proposed a novel Feature Purification Network (FP-Net) to improve the representation for text classification. The method is based on feature projection. The proposed model uses two sub-networks, one for identifying common features that are not discriminative for classification, and the other for feature projection that projects the traditional features to the orthogonal direction of the common features. To the best of our knowledge, this is the first method that uses feature projection to improve text classification. Through a large number of comparative experiments, we showed the effectiveness of the proposed feature projection method.

Our current method is designed only for traditional text classification methods such as LSTM, CNN, and Transformer. In our future work, we will consider extending it to graph-based methods such as GCN for graph data, and to generation-based methods such as GAN for adversarial learning.

Acknowledgments

The project was funded by Peking University.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yonatan Belinkov, Adam Poliak, Stuart M Shieber, Benjamin Van Durme, and Alexander M Rush. 2019. On adversarial removal of hypothesis-only bias in natural language inference. *arXiv preprint arXiv:1907.04389*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

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.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

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*.

Kawin Ethayarajh. 2018. Unsupervised random walk sentence embeddings: A strong but simple baseline. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 91–100.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 171–175. Association for Computational Linguistics.

Yaroslav Ganin and Victor Lempitsky. 2014. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1681–1691.

Samuel R. Bowman Jason Phang, Thibault Févry. 2019. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv:1811.01088 [cs.CL].*

Peng Jin, Yue Zhang, Xingyuan Chen, and Yunqing Xia. 2016. Bag-of-embeddings for text classification. In *IJCAI*, volume 16, pages 2824–2830.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics.*

Christopher D. Manning Quoc V. Le Kevin Clark, Minh-Thang Luong. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2002 conference on Empirical methods in natural language processing*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Yukun Ma, Haiyun Peng, and Erik Cambria. 2018. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2015. Natural language inference by tree-based convolution and heuristic matching. *arXiv preprint arXiv:1512.08422*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Dominik Scherer, Andreas Müller, and Sven Behnke. 2010. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, pages 92–101. Springer.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Y. Tamaazousti, H. Le Borgne, C. Hudelot, M. E. A. Seddik, and M. Tamaazousti. 2018. Learning more universal representations for transfer-learning. *arXiv:1712.09708v5 [cs.CV]*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Baoxin Wang. 2018. Disconnected recurrent neural networks for text categorization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2311–2320.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Yiren Wang and Fei Tian. 2016. Recurrent residual learning for sequence classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 938–943.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

Kai Zhang, Hefu Zhang, Qi Liu, Hongke Zhao, Hengshu Zhu, and Enhong Chen. 2019. Interactive attention transfer network for cross-domain sentiment classification.