How to make your BERT model an xBERT in multitask learning?

Stanford CS224N Default Project
Mentor: Timothy Dai

Xingshuo Xiao
Institute for Computational and Mathematical Engineering
Stanford University
xingshuo@stanford.edu

Abstract

Multitask learning (MTL) is popular in natural language processing (NLP) due to its ability to leverage shared knowledge across related tasks. However, its effectiveness compared to training tasks individually remains unclear. In addition, fine-tuning pre-trained models for MTL poses challenges such as overfitting and balancing task-specific and overall performance. This project investigates the performance of MTL models and various regularization techniques to mitigate overfitting using a BERT-based architecture for three NLP tasks: sentiment analysis, paraphrase detection, and semantic textual similarity (STS). Our findings show that while MTL models can benefit from shared knowledge, they do not outperform task-specific models. L2 regularization is more effective than SMART regularization in improving model generalization and robustness. Due to the limited computational resources, future work could explore more advanced architectures, further pre-training on BERT, and extensive hyperparameter tuning to further examine the MTL effectiveness.

1 Introduction

Multitask learning (MTL) has become popular in natural language processing (NLP) because it can leverages shared knowledge across related tasks to improve overall performance. However, it is still unclear whether multitask learning is more effective than training each task separately. Additionally, fine-tuning pre-trained models for MTL is challenging due to overfitting and the need to balance task-specific and overall performance.

In this project, we aim to investigate the effectiveness of MTL and regularization techniques to alleviate overfitting issues. We conduct experiments to compare the performance of MTL models to task-specific models trained individually for each task. By analyzing results from three distinct tasks - sentiment analysis, paraphrase detection, and semantic textual similarity (STS) - we hope to identify how to optimize MTL performance and understand the trade-offs between individual and average performance. For multitask learning, overfitting is usually a major concern. To address this issue, we implement various regularization techniques, including L2 regularization, dropout, and SMART regularization (Jiang et al., 2020).

2 Related Work

2.1 BERT

BERT is a pre-trained language representation model proposed by Devlin et al.. Compared to previous models, BERT is able to pre-train bidirectional encoders by jointly conditioning on both left and right context in all layers. BERT uses a masked language model (MLM) and next sentence prediction

(NSP) tasks to pre-train the model, which allows BERT to understand the semantic context in a sentence.

With the pre-trained weights, BERT can be fine-tuned with minimal additional architecture for various NLP tasks, including but not limited to text classification, named entity recognition, and question answering. The simplest way is to add just one additional output layer. BERT represents the strength of transfer learning from pre-trained language models. Additional improvements in efficiency and computational costs have been made to BERT, such as RoBERTa (Liu et al., 2019b), DistilBERT (Sanh et al., 2020), and ALBERT (Lan et al., 2020). Models are also built based on BERT to tackle NLP tasks in specific domains, including FinBERT (Araci, 2019) for financial sentiment analysis and BioBERT (Lee et al., 2019) for biomedical text mining.

2.2 Multitask Learning

MTL models train models for multiple related tasks at the same time. The shared knowledge across tasks can help improve the overall performance of the model. Liu et al. proposed a multitask deep neural network incorporating BERT, which achieves state-of-the-art performance on multiple NLP tasks. The architectures of MTL models depend on the tasks and the base model. For highly related tasks, weight sharing can be applied to improve the efficiency of learning and avoid overfitting for a specific task (Prellberg and Kramer, 2020). Task-specific layers can be added to the base model to capture task-specific features. They can help make the MTL model more flexible and robust to different tasks.

2.3 Regularization of Multitask Learning

Regularization techniques in MTL have been extensively studied to address overfitting and the difficulty in managing complex relationships between tasks. Some works have proposed innovative methods to enhance the generalization and robustness of MTL models. Meshgi et al. explore the problem that fine-tuning on one task might negatively affect the performance on others. They propose using task uncertainty to weigh the impact of shared feature changes on different tasks, and therefore prevent overfitting and over-generalization. Liu et al. present a weighted regularized MTL framwork to reduce the impact of outlier tasks and achieves better overall performance. Jiang et al. propose a novel learning framework for tine-tuning pre-trained language models called SMART to improve robustness and generalization of MTL models.

3 Approach

3.1 Model Architecture

The BERT architecture is described in the original BERT paper (Devlin et al., 2019). Our MultitaskBERT is built on the base BERT model with three task-specific heads for sentiment analysis, paraphrase detection, and STS tasks. The BERT model architecture is shown in Figure 1.

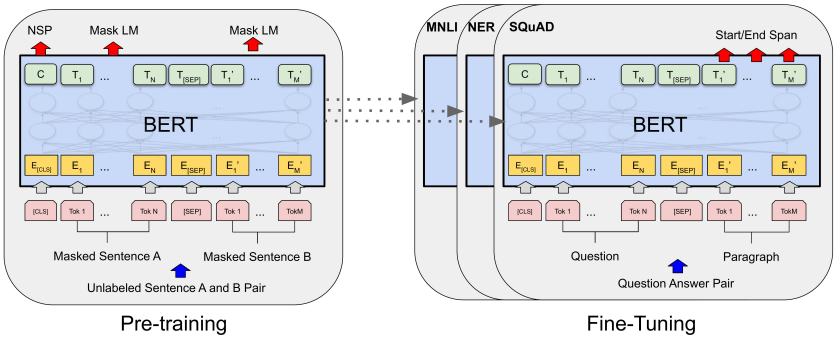

Figure 1: BERT Model Architecture.

3.2 Baselines

Our baseline model is the minBERT model from the sentiment analysis task in the CS224N default project. A dropout layer is added after the BERT model to prevent overfitting on the embeddings. Three task-specific heads with linear layers are added to the dropout layer to implement the three downstream tasks. We consider two baselines: (1) pre-trained embeddings (provided by CS224N default project) with frozen BERT parameters and (2) fine-tuned BERT model parameters. In both baselines, the tasks are trained sequentially in each epoch.

3.3 DNN and Sharing Weights

Unlike the baselines, we add more layers in each task-specific head to capture task-specific features. The model architecture is shown in Figure 2. Since the paraphrase detection and STS tasks are related, we share the weights of the first two layers in the task-specific heads. The DNN layers are connected with GELU (Hendrycks and Gimpel, 2023).

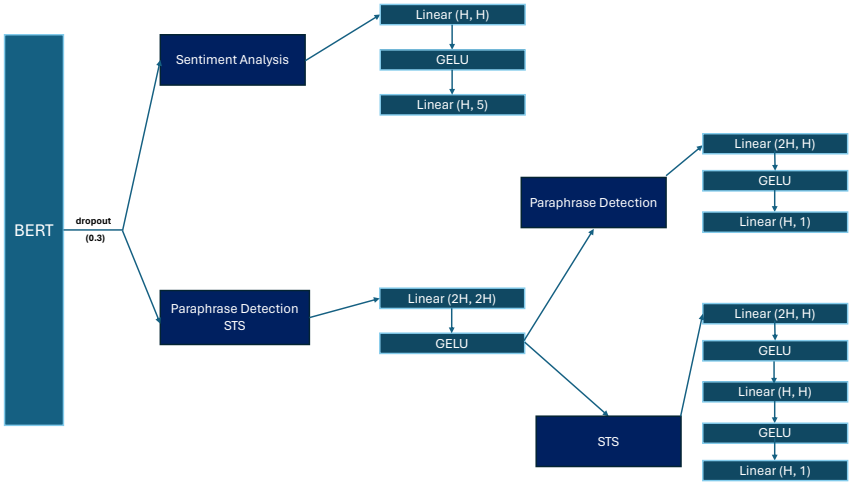

Figure 2: Multitask BERT Model Architecture.

3.4 Multitask Learning (MTL) with Gradient Surgery (GS)

MTL allows the model to learn from tasks simultaneously by updating gradients using the sum of the losses from each task. However, for different tasks, the gradients might be conflicting, leading to slow learning or poor performance. To address this issue, we implement the gradient surgery (GS) technique (Yu et al., 2020) using PCGrad (Tseng, 2020). The GS technique projects a conflicting gradient g_i onto the normal plane of another gradient g_j:

$$g_i \leftarrow g_i - \left(\frac{g_i \cdot g_j}{\|g_j\|^2} \right) g_j \tag{1}$$

During the training, we sample the data from each task in one batch. We explore two approaches handling the differences in data sizes of tasks:

- using the same batch size for all tasks (fixed `batch_size`)
- adjusting the batch size for each task based on the data size, with the smallest dataset (STS) determining the number of batches (adjusted `batch_size`)

With the fixed `batch_size` approach, we are unable to exploit the training data from the larger datasets, while with the adjusted `batch_size` approach, there is a challenge with high memory costs as the paraphrase dataset is significantly larger. Due to the limited computational resources, we only use 18,120 samples from the paraphrase task training data.

3.5 Regularization

3.5.1 SMART Regularization

SMART regularization is implemented using the code written by Liu et al.. Given the model $f(\cdot;\theta)$ and n observations $\{x_i, y_i\}_{i=1}^{n}$ from the task, the method solves the optimization objective for fine-tuning:

$$\min_{\theta} \quad \mathcal{L}\left(f\left(x_i;\theta\right), y_i\right) + \lambda_s \mathcal{R}_s\left(\theta\right) \tag{2}$$

with the terms specified in Appendix A.

3.5.2 L2 Regularization

L2 regularization is implemented through adding a weight decay term to the AdamW optimizer.

3.6 Ensemble

We ensemble the predictions from multiple models to improve the overall performance. For the classification tasks, we use the majority voting strategy to combine the predictions from the models. For the regression task, we average the predicted scores from the models.

3.7 Task-specific Model

As a reference to MTL models, we train task-specific models for each task. The model architecture is the same as the MTL model, but the model is trained separately for each task.

4 Experiments

4.1 Data

We used the 3 datasets provided by the CS224N default project:

- **Stanford Sentiment Treebank (SST)** (Socher et al., 2013): The SST dataset consists of 11,855 movie review sentences labeled as negative, somewhat negative, neutral, somewhat positive, and positive. The dataset includes 215,154 unique phrases and labeled by 3 judges.
- **Quora Dataset (QQP)** [1]: The Quora dataset consists of 404,298 question pairs with binary labels indicating whether they are paraphrases of each other.
- **SemEval STS Benchmark Dataset** (Agirre et al., 2013): The SemEval STS Benchmark dataset consists of 8,628 sentence pairs with similarity ratings from 0 (unrelated) to 5 (equivalent meaning).

All datasets are preprocessed and split into `train`, `dev`, and `test` sets with a ratio of 0.7, 0.1, and 0.2, respectively.

4.2 Evaluation method

Accuracy is used as the primary evaluation metric for sentiment analysis and paraphrase detection. For senmatic texual similarity task, Pearson correlation of the true similarity scores against the predicted scores is used as the evaluation metric (Agirre et al., 2013). The overall performance score is calculated by averaging the metrics for each task. To keep the scores consistent, we normalize the STS correlation scores to the range of 0 to 1 before calculating the average.

Qualitative analysis is also conducted to understand the model's performance and any existing patterns. Specific examples where the model achieves or fails the goal in each task are picked from the `dev` set.

[1] `https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs`

4.3 Experimental details

All experiments used AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon =$1e-6), a hidden dropout rate of 0.3, and a weight decay of 0.01. The learning rate is set to 1e-3 for running pretrained BERT model and 1e-5 for fine-tuning. The numbed of epochs is 10. The batch size is 8 for most models, except for the multi-task fine-tuning model with adjusted batch sizes (STS: 8, SST: 12, QQP: 12). During the training process, cross-entropy loss is used for SST and Quora tasks, and mean squared error loss is used for the STS task. All experiments are conducted on a single NVIDIA T4 GPU on GCP.

Here is a table of models and their descriptions:

Table 1: Model Descriptions.

Model #	Description
1	Baseline (Pre-trained)
2	Baseline (Fine-tune)
3	Weights sharing + L2 (Fine-tune)
4	Weights sharing + SMART (Fine-tune)
5	MTL (Fixed batch size, Fine-tune)
6	MTL (Adjusted batch size, Fine-tune)
7	Ensemble 1 (Model 3 + Model 4 + Model 5)
8	Ensemble 2 (Model 2 + Model 3 + Model 6)
9	Reference: Task-specific (Fine-tune)

4.4 Results

The accuracy and Pearson correlation scores on the dev and test leaderboard are reported in Table 2. The dev results for each model are shown in Table 3.

Table 2: Leaderboard Results.

Leaderboard	Overall	SST Acc.	Para Acc.	STS Corr.
Dev	0.668	0.500	0.755	0.499
Test	0.677	0.521	0.755	0.510

Adding sharing weights and L2 regularization to the baseline model improves the overall performance on the `dev` set. This is expected as the model can capture more task-specific features and prevent overfitting on the training data.

The MTL models do not perform on all three tasks as well as baseline and the task-specific models. This might because we only use a simple DNN structure for the task-specific heads. For the two batch size approaches, the results are almost the same. However, we are unable to conclude that both approaches are equally effective due to limited use of QQP data in both ways.

The ensemble models show better performance than the MTL models, which suggests that ensembling can help improve the overall performance of the model.

Compared to L2 regularization, SMART regularization does not show significant improvement in the model performance. In our experiments, we only implemented smoothness-inducing adversarial regularization for SMART. The performance of SMART is not optimized without Bergman proximal point optimization. Also, as shown in Figure 3, for the tasks with less training data, regularization techniques do not show significant improvement in overfitting.

Lastly, we could tell that the paraphrase detection task has the best performance among the three tasks. This might be attributed to rich amount of training data and the nature of binary classification that is easier to train.

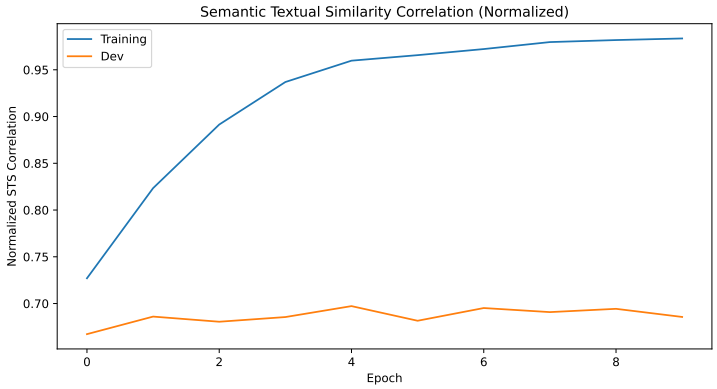

Figure 3: Model 4 - Normalized STS correlation on `train` and `dev`.

Table 3: Dev Results.

Model	Overall	SST Acc.	Para Acc.	STS Corr.
Model 1	0.564	0.401	0.661	0.259
Model 2	0.640	0.444	0.796	0.361
Model 3	0.667	0.365	**0.847**	**0.575**
Model 4	0.625	0.465	0.714	0.390
Model 5	0.570	0.391	0.682	0.274
Model 6	0.570	0.391	0.682	0.274
Model 7	0.646	0.496	0.747	0.389
Model 8	0.668	**0.500**	0.755	0.499
Model 9	0.789	0.532	0.893	0.881

5 Analysis

We analyze the results from Model 6 with some specific examples of the three downstream tasks to understand the model's behavior.

5.1 Sentiment Analysis

When the input sentences contain clear and unambiguous words that indicate sentiment, such as "best", "terrible" and "engaging", the MTL model performs well. Also, the model is able to identify nuanced negative sentiments in sentences such as Example 1 that are simple-structured and straightforward.

Example 1

Input Sentence:
"no one goes unindicted here, which is probably for the best."
Predicted Sentiment: 2 (somewhat negative)
Actual Sentiment: 2 (somewhat negative)

However, the model struggles when the input sentences have complex grammatical structures, idiomatic expressions or neutral descriptions. For example, the model tends to overestimate positive sentiments when some positive adjectives are used.

Example 2

Input Sentence:
"it 's a lovely film with lovely performances by buy and accorsi."
Predicted Sentiment: 4 (somewhat positive)
Actual Sentiment: 3 (neutral)

This demonstrates that our MTL model can successfully understand the "sentiment" of adjectives but struggles to capture the overall sentiment of the sentence especially when sarcasm and more complicated emotional expressions are involved.

Furthermore, in our predictions, we notice that the model tends to give predictions that is only 1 class away from the truth. This suggests that the model gives predictions that are close to the true sentiment but not exactly the same.

5.2 Paraphrase Detection

For the paraphrase detection task, the MTL model performs well to detect the not paraphrased inputs, even if they share a common subject matter (e.g. Example 3). However, the model struggles with sentences that have minor lexical differences but are semantically equivalent (e.g. Example 4). Furthermore, the model tends to make incorrect classifications when the sentences vary significantly in structure but convey the same meaning. Sometime, the model fails to recognize paraphrases that use antonyms. That might be due to the lack of training data pattern.

Example 3

Input Sentence 1:
"what are some great ways to generate passive income in india?"
Input Sentence 2:
"can a full time employee be successful in investing on stock markets and other potential investments to generate passive income?"
Predicted Paraphrase: 0 (Not paraphrased)
Actual Paraphrase: 0 (Not paraphrased)

Example 4

Input Sentence 1:
"what is shaoxing vinegar?"
Input Sentence 2:
"what is shaoxing vinegar used for?"
Predicted Paraphrase: 0 (Not paraphrased)
Actual Paraphrase: 1 (Paraphrased)

It is important to note that the data is labeled by human judges and might make mistakes. For instance, Example 4 is labeled as paraphrased, but the sentences are not exactly the same in meaning. Some people might even have different opinions whether the sentences are paraphrased or not. Therefore, for those cases, it is reasonable that the model predictions are not fully aligned with the data labels.

5.3 Semantic Textual Similarity

The model performs well when the input sentences share similar phrases or have slight variations in wording. It also shows capabilities in identifying sentences that describe the same action or event with minor contextual differences. Most cases the model fails (difference between predicted and actual scores is greater than 1) have input sentences that describe significantly different actions but have similar sentence structure (e.g. Example 5). This leads to overestimation of the similarity between the sentences. It suggests that the model might have learned the sentence structure patterns but failed to capture the semantic meaning of the sentences.

Example 5

Input Sentence 1:
"a man is mowing a lawn."
Input Sentence 2:
"a woman is cutting a lemon."
Predicted Similarity Score: 4.1
Actual Similarity Score: 0.2

6 Conclusion

In this project, we explored the effectiveness of multitask learning using a BERT model architecture for three distinct NLP tasks. Our findings showed that while MTL models can leverage shared knowledge across tasks, they did not outperform task-specific models and the baseline model training tasks sequentially. However, incorporating regularization techniques such as L2 regularization and SMART regularization improved model generalization and robustness, though these improvements were not uniform across all tasks. Ensembled models, which combined the predictions from multiple models, significantly boosted performance, demonstrating the utility of ensembling in enhancing overall model accuracy.

However, there are still limitations in our work investigating the effectiveness of MTL. Computational resource constraints restricted our ability to fully exploit the larger datasets, particularly for the paraphrase detection task. This potentially limited our MTL models' potential. Additionally, the simple DNN structure used for task-specific heads might not have been sufficient to capture complex task-specific features, and overfitting remained a concern, especially for tasks with less training data.

Future work could involve exploring more sophisticated task-specific head architectures such as projected attention layers (Stickland and Murray, 2019) to capture more complex task-specific features. Further pre-training the BERT base model on domain-specific data could also improve the model's performance on specific tasks.

7 Ethics Statement

In this project, some ethical challenges related to the datasets need to be considered. For example, the paraphrase detection data comes from the QQP dataset, which contains user-generated content and may include personal or sensitive information. This would raise privacy concerns for the users. In addition, the SST dataset was labeled by 3 human judges, which might impose biases in the training data, leading to biased model predictions that reinforce harmful societal prejudices.

To mitigate these risks, there are three possible approaches: pre-processing, in-processing, and post-processing. For pre-processing, we can examine the data and remove any sensitive information. For in-processing, regularzation techniques can be used to reduce the model's reliance on the training data and prevent learning the biases from the training data. For post-processing, we can analyze the model predictions thoroughly and identify any biases or harmful outputs.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dan Hendrycks and Kevin Gimpel. 2023. Gaussian error linear units (gelus).

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding.

Xiaodong Liu, Yu Wang, Jianshu Ji, Hao Cheng, Xueyun Zhu, Emmanuel Awa, Pengcheng He, Weizhu Chen, Hoifung Poon, Guihong Cao, and Jianfeng Gao. 2020. The microsoft toolkit of multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:2002.07972*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach.

Yintao Liu, Anqi Wu, Dong Guo, Ke-Thia Yao, and Cauligi S. Raghavendra. 2013. Weighted task regularization for multitask learning. In *2013 IEEE 13th International Conference on Data Mining Workshops*, pages 399–406.

Kourosh Meshgi, Maryam Sadat Mirzaei, and Satoshi Sekine. 2022. Uncertainty regularized multi-task learning. In *Proceedings of the 12th Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis*, pages 78–88.

Jonas Prellberg and Oliver Kramer. 2020. Learned weight sharing for deep multi-task learning by natural evolution strategy and stochastic gradient descent.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning.

Wei-Cheng Tseng. 2020. Weichengtseng/pytorch-pcgrad.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning.

A SMART Regularization

The loss function is defined as:

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} l\left(f\left(x_i;\theta\right), y_i\right) \tag{3}$$

where l is the loss function for the specific task, λ_s is a positive tuning parameters, and $\mathcal{R}_s$ is the smoothness-inducing regularizer:

$$\mathcal{R}_s(\theta) = \frac{1}{n}\sum_{i=1}^{n} \max_{\|\tilde{x_i} - x_i\|_p \leq \epsilon} l_s\left(f\left(\tilde{x_i}\right), f\left(x_i;\theta\right)\right) \tag{4}$$

where $\epsilon > 0$ is a tuning parameter. For classification tasks, l_s is defined as:

$$l_s(P, Q) = \mathcal{D}_{KL}\left(P \| Q\right) + \mathcal{D}_{KL}\left(Q \| P\right) \tag{5}$$

and for regression tasks, l_s is defined as:

$$l_s(P, Q) = \left(P - Q\right)^2 \tag{6}$$