Single Model Ensemble using Pseudo-Tags and Distinct Vectors

Ryosuke Kuwabara[1], **Jun Suzuki**[23], **Hideki Nakayama**[1]
[1] The University of Tokyo, [2] Tohoku University, [3] RIKEN
{kuwabara, nakayama}@nlab.ci.i.u-tokyo.ac.jp
jun.suzuki@ecei.tohoku.ac.jp

Abstract

Model ensemble techniques often increase task performance in neural networks; however, they require increased time, memory, and management effort. In this study, we propose a novel method that replicates the effects of a model ensemble with a single model. Our approach creates K-virtual models within a single parameter space using K-distinct pseudo-tags and K-distinct vectors. Experiments on text classification and sequence labeling tasks on several datasets demonstrate that our method emulates or outperforms a traditional model ensemble with $1/K$-times fewer parameters.

1 Introduction

A model ensemble is a promising technique for increasing the performance of neural network models (Lars. and Peter., 1990; Anders and Jesper, 1994). This method combines the outputs of multiple models that are individually trained using the same training data. Recent submissions to natural language processing(NLP) competitions are primarily composed of neural network ensembles (Bojar et al., 2018; Barrault et al., 2019). Despite its effectiveness, a model ensemble is costly. Because it handles multiple models, it requires increased time for training and inference, increased memory, and greater management effort. Therefore, the model ensemble technique cannot always be applied to real systems, as many systems, such as edge devices, must work with limited computational resources.

In this study, we propose a novel method that replicates the effects of the ensemble technique with a single model. Following the principle that aggregating multiple models improves performance, we create multiple virtual models in a shared space. Our method virtually inflates the training data K times with K-distinct pseudo-tags appended to all input data. It also incorporates K-distinct vectors, which correspond to pseudo-tags. Each pseudo-tag $k \in \{1, \ldots, K\}$ is attached to the beginning of the input sentence, and the k-th vector is added to the embedding vectors for all tokens in the input sentence. Fig. 1 presents a brief overview of our proposed method. Intuitively, this operation allows the model to shift the embedding of the same data to the k-th designated subspace and can be interpreted as explicitly creating K virtual models in a shared space. We thus expect to obtain the same (or similar) effects as the ensemble technique composed of K models with our K virtual models generated from a single model.

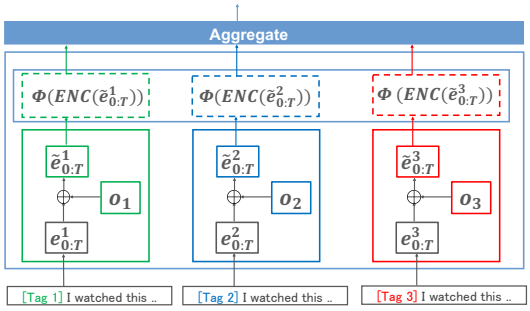

Figure 1: Overview of our proposed method. A single model processes the same input with distinct pseudo-tags. Each pseudo-tag defines the k-th virtual model, and the corresponding vector $\boldsymbol{o}_k$ is added to the embedding. Thus, the model function of a singe model $\phi\,(\mathrm{ENC}(\cdot))$ generates different outputs.

Experiments in text classification and sequence labeling tasks reveal that our method outperforms single models in all settings with the same parameter size. Moreover, our technique emulates or surpasses the normal ensemble with $1/K$-times fewer parameters on several datasets.

2 Related Work

The neural network ensemble is a widely studied method (Lars. and Peter., 1990; Anders and Jesper,

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 3006–3013

1994; Hashem, 1994; Opitz and Shavlik, 1996); however studies have focused mainly on improving performance while ignoring cost, such as computational cost, memory space, and management cost.

Several methods have overcome the shortcomings of traditional ensemble techniques. For training Snapshot Ensembles, (Huang et al., 2017) used a single model to construct multiple models by converging into multiple local minima along the optimization path. For inference distillation, (Hinton et al., 2015) transferred the knowledge of the ensemble model into a single model. These methods use multiple models either during training or inference, which partially solves the negative effects of the traditional ensemble.

The incorporation of pseudo-tags is a standard technique widely used in the NLP community, (Rico et al., 2016; Melvin et al., 2017). However, to the best of our knowledge, our approach is the first attempt to incorporate pseudo-tags as an identification marker of virtual models within a single model.

The most similar approach to ours is dropout (Srivastava et al., 2014), which stochastically omits each hidden unit during each mini-batch, and in which all units are utilized for inference. Huang et al. (2017) interpreted this technique as implicitly using an exponential number of virtual models within the same network. As opposed to dropout, our method explicitly utilizes virtual models with a shared parameter, which is as discussed in Section 5, complementary to dropout.

3 Base Encoder Model

The target tasks of this study are text classification and sequence labeling. The input is a sequence of tokens (i.e., a sentence). Here, $\boldsymbol{x}_t$ denotes the one-hot vector of the t-th token in the input. Let $\boldsymbol{E} \in \mathbb{R}^{D \times |\mathcal{V}|}$ be the embedding matrices where D is the dimension of the embedding vectors and $\mathcal{V}$ is the vocabulary of the input.

We obtain the embedding vector $\boldsymbol{e}_t$ at position t by $\boldsymbol{e}_t = \boldsymbol{E}\boldsymbol{x}_t$. Here, we introduce the notation $\boldsymbol{e}_{1:T}$ to represent the list of vectors $(\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_T)$ that correspond to the input sentence, where T is the number of tokens in the input. Given $\boldsymbol{e}_{1:T}$, the feature (or hidden) vectors $\boldsymbol{h}_t \in \mathbb{R}^H$ for all $t \in \{1, \ldots, T\}$ are computed as an encoder neural network $\textsc{enc}(\cdot)$, where H denotes the dimensions of the feature vector. Namely,

$$\boldsymbol{h}_{1:T} = \textsc{enc}\left(\boldsymbol{e}_{1:T}\right). \tag{1}$$

Finally, the output $\widehat{\boldsymbol{y}}$ given input $\boldsymbol{x}_{1:T}$ is estimated as $\widehat{\boldsymbol{y}} = \phi\left(\boldsymbol{h}_{1:T}\right)$ where $\phi\left(\cdot\right)$ represents the task dependent function (e.g., a softmax function for text classification and a conditional random field layer for sequence labeling). It should be noted that the form of the output $\widehat{\boldsymbol{y}}$ differs depending on the target task.

4 Single Model Ensemble using Pseudo-Tags and Distinct Vectors

In this section, we introduce the proposed method, which we refer to as SINGLEENS. Fig. 1 presents an overview of the method. The main principle of this approach is to create different virtual models within a single model.

We incorporate **pseudo-tags** and **predefined distinct vectors**. For the pseudo-tags, we add special tokens $\{\boldsymbol{\ell}_k\}_{k=1}^K$ to the input vocabulary, where hyper-parameter K represents the number of virtual models. For the predefined distinct vectors, we leverage mutually orthogonal vectors $\{\boldsymbol{o}_k\}_{k=1}^K$, where the orthogonality condition requires satisfying $\boldsymbol{o}_k \cdot \boldsymbol{o}_{k'} \simeq 0$ for all (k, k') when $k \neq k'$.

Finally, we assume that all input sentences start from one of the pseudo-tags. We then add the corresponding orthogonal vector $\boldsymbol{o}_k$ of the attached pseudo-tag $\boldsymbol{\ell}_k$ to the embedding vectors at all positions. The new embedding vector $\tilde{\boldsymbol{e}}_{0:T}$ is written in the following form:

$$\tilde{\boldsymbol{e}}_{0:T}^{(k)} = (\boldsymbol{\ell}_k, \boldsymbol{e}_1 + \boldsymbol{o}_k, \boldsymbol{e}_2 + \boldsymbol{o}_k, \ldots, \boldsymbol{e}_T + \boldsymbol{o}_k). \tag{2}$$

We substitute $\boldsymbol{e}_{1:T}$ in Eq. 1 by $\tilde{\boldsymbol{e}}_{0:T}^{(k)}$ in the proposed method.

An intuitive explanation of the role of pseudo-tags is to allow a single model to explicitly recognize differences in homogeneous input, while the purpose of orthogonal vectors is to linearly shift the embedding to the virtual model's designated direction. Therefore, by combining these elements, we believe that we can define virtual models within a single model and effectively use the local space for each virtual model. Aggregating these virtual models can then result in imitation of ensemble.

5 Experiments

To evaluate the effectiveness of our method, we conducted experiments on two tasks: text classification and sequence labeling. We used the IMDB (Andrew et al., 2011), Rotten (Bo and Lillian,

Dataset	Model	Method	# params	Accuracy
IMDB	TFM: GLOVE	SINGLE	12 M	87.03
		1/K ENS	14 M	81.93 (−5.10)
		SINGLEENS	12 M	87.30 (+0.27)
		NORMALENS	108 M	**87.67** (+0.64)
	TFM: BERT	SINGLE	400 M	91.99
		1/K ENS	1000 M	90.63 (−1.36)
		SINGLEENS	400 M	**92.91** (+0.92)
		NORMALENS	3600 M	92.75 (+0.76)
Rotten	TFM: BERT	SINGLE	400 M	81.75
		1/K ENS	1000 M	82.67 (+0.92)
		SINGLEENS	400 M	**85.01** (+3.26)
		NORMALENS	3600 M	82.57 (+0.82)
RCV1	TFM: BERT	SINGLE	400 M	87.18
		1/K ENS	1000 M	80.27 (−6.91)
		SINGLEENS	400 M	89.16 (+1.98)
		NORMALENS	3600 M	**90.01** (+2.83)

Table 1: Test accuracy and parameter size for text classification tasks. Our method, **SINGLEENS**, outperformed SINGLE and 1/K ENS on all datasets. Most notably, **SINGLEENS** surpassed NORMALENS on IMDB and Rotten with 1/9 fewer parameters.

Dataset	Model	Method	# params	F1 Score
CoNLL 2003	TFM: ELMO	SINGLE	100 M	91.93
		1/K ENS	150 M	91.65 (−0.28)
		SINGLEENS	100 M	92.37 (+0.44)
		NORMALENS	900 M	**92.86** (+0.93)
CoNLL 2000	TFM: ELMO	SINGLE	100 M	96.42
		1/K ENS	150 M	95.67 (−0.75)
		SINGLEENS	100 M	96.56 (+0.14)
		NORMALENS	900 M	**96.67** (+0.25)

Table 2: Test F1 score and parameter size for sequence labeling tasks. Similarly to NORMALENS, **SINGLEENS** improved the score even at high performance levels.

2005), and RCV1 (Yiming et al., 2004) datasets for text classification, and the CoNLL-2003 (Sang and Meulder, 2003) and CoNLL-2000 datasets (Sang and Sabine, 2000) for sequence labeling.

We used the Transformer model (Vaswani et al., 2017) as the base model for all experiments, and its token vector representations were then empowered by pretrained vectors of GloVe, (Jeffrey et al., 2014), BERT (Devlin et al., 2018), or ELMo (Matthew et al., 2018). The models are referred to as TFM:GLOVE, TFM:BERT, and TFM:ELMO, respectively.[1] For TFM:BERT, we incorporated the feature (or hidden) vectors of the final layer in the BERT model as the embedding vectors while adopting drop-net technique (Zhu et al., 2020). All the models have dropout layers to assess the complementarity of our method and dropout.

We compared our method (**SINGLEENS**) to a single model (**SINGLE**), a normal ensemble (**NORMALENS**), and a normal ensemble in which each component has approximately $1/K$ parameters[2] (**1/K ENS**).[3] Although other ensemble-like methods discussed in Section 2 could have been compared (e.g., snapshot ensemble, knowledge distillation, or dropout during testing to generate predictions and aggregate them), they are imitations of a normal ensemble, and we assumed that the results of a normal ensemble were upper-bound. We used $K = 9$ for reporting the primary results of NORMALENS, 1/K ENS, and SINGLEENS. We thus prepared nine pseudo-tags $\{\boldsymbol{\ell}_k\}_{k=1}^{9}$ in the same training (trainable) and initialization manner as other embeddings. We created untrainable distinct vectors $\{\boldsymbol{o}_k\}_{k=1}^{9}$ using the implementation by Saxe et al. (2013) that was prepared in PyTorch's default function, `torch.nn.init.orthogonal`. We empirically determined the correct scaling for the distinct vectors as 1 out of 1, 3, 5, 10, 30, 50, 100, and the scale that was closest to the model's embedding vectors. We obtained the final predictions of K ensemble models by averaging and voting the outputs of individual models for text classification and sequence labeling, respectively. The results were obtained by the averaging five distinct runs with different random seeds.

5.1 Evaluation of text classification

Data We followed the settings used in the implementation by Kiyono et al. (2018) for data partition.[4] Our method, SINGLEENS inflates the training data by K times. During the inflation, the k-th subset is sampled by bootstrapping (Efron and Tibshirani, 1993) with the corresponding k-th pseudo-tag. For NORMALENS and 1/K ENS, we attempted both bootstrapping and normal sampling, and a higher score was reported.

Results Table 1 presents the overall results evaluated in terms of accuracy. For both TFM:GLOVE and TFM:BERT, SINGLEENS outperformed SINGLE with the same parameter size. In our experiments, SINGLEENS achieved the best scores on IMDB and Rotten with TFM:BERT; it recorded 92.91% and 85.01%, which was higher than NORMALENS by 0.16 and 2.44, respectively with 89% fewer parameters. The standard deviation of the results for the IMDB dataset was, 0.69 and 0.14

[1] See Appendix A for detailed experimental settings.

[2] Because BERT requires a fixed number of parameters, we did not reduce the parameters accurately for 1/K TFM:BERT.

[3] See Appendix A for detailed experimental settings.

[4] See Appendix B for data statistics.

Setting	IMDB Accuracy	CoNLL-2003 F1 Score
SINGLE	91.99	91.93
1) Only pseudo-tags	89.84	92.20
2) Random distinct vectors	92.06	92.21
3) Random noise	92.38	92.32
SINGLEENS	**92.91**	**92.37**

Table 3: Comparison of proposed method (pseudo-tags + corresponding distinct vectors) with other settings. Pseudo-tags and distinct vectors appear to complement each other.

Setting	IMDB Accuracy	CoNLL-2003 F1 Score
SINGLE	91.99	91.93
1) Emb (SINGLEENS)	**92.91**	92.37
2) Hidden	90.68	**92.45**
1) + 2)	92.64	92.19

Table 4: Test metrics on IMDB and CoNLL-2003 with the pattern of three vector addition operations. Adding distinct vectors to only embeddings is the best or second best approach.

for SINGLE and SINGLEENS, respectively, for TFM:GLOVE, and 0.34 and 0.11, respectively, for TFM:BERT. These results support the claim that explicit operations for defining K virtual models have a significant effect for a single model and are complementary to normal dropout. Through the series of experiments, we observed that the number of iterations of SINGLEENS was 1.0 ˜1.5 times greater than that of SINGLE.

5.2 Evaluation of sequence labeling

Data We followed the instructions of the task settings used in CoNLL-2000 and CoNLL-2003.[5] We inflated the training data by nine times for SINGLEENS, and normal sampling was used for NORMALENS and 1/K ENS. Because bootstrapping was not effective for the task, the results were omitted.

Results As displayed in Table 2, SINGLEENS surpassed SINGLE by 0.44 and 0.14 on CoNLL-2003 and CoNLL-2000, respectively, for TFM:ELMO with the same parameter size. However, NORMALENS produced the best results in this setting. The standard deviations of the single model and our methods were 0.08 and 0.05, respectively, on CoNLL-2000. Through the series of experiments, we observed that the number of iterations of SINGLEENS was 1.0 ˜1.5 times greater than that of SINGLE.

6 Analysis

In this section, we investigate the properties of our proposed method. Unless otherwise specified, we use TFM:BERT and TFM:ELMO on IMDB and CoNLL-2003 for the analysis.

Significance of pseudo-tags and distinct vectors To assess the significance of using both pseudo-tags and distinct vectors, we conducted an ablation study of our method, SINGLEENS. We compared our method with the following three settings: 1) Only pseudo-tags, 2) Random distinct vectors, and 3) Random noise. In detail, the first setting (Only pseudo-tags) attached the pseudo-tags to the input without adding the corresponding distinct vectors. The second setting (Random distinct vectors) randomly shuffles the correspondence between the distinct vectors and pseudo-tags in every iteration during the training. Additionally, the third setting (Random noise) adds random vectors as the replacement of the distinct vectors to clarify whether the effect of incorporating distinct vectors is essentially identical to the random noise injection techniques or explicit definition of virtual models in a single model.

Table 3 shows the results of the ablation study. This table indicates that using both pseudo-tags and distinct vectors, which matches the setting of SINGLEENS, leads to the best performance, while the effect is limited or negative if we use pseudo-tags alone or distinct vectors and pseudo-tags without correspondence. Thus, this observation explains that the increase in performance can be attributed to the combinatorial use of pseudo-tags and distinct vectors, and not merely data augmentation.

We can also observe from Table 3 that the performance of SINGLEENS was higher than that of 3) Random noise. Note that the additional vectors by SINGLEENS are fixed in a small number K while those by Random noise are a large number of different vectors. Therefore, this observation supports our claim that the explicit definition of virtual models by distinct vectors has substantial positive effects that are mostly irrelevant to the effect of the random noise. This observation also supports the assumption that SINGLEENS is complementary to dropout. Dropout randomly uses sub-networks by stochastically omitting each hidden unit, which can be interpreted as a variant of Random noise.

[5] The statistics of the datasets are presented in Appendix B.

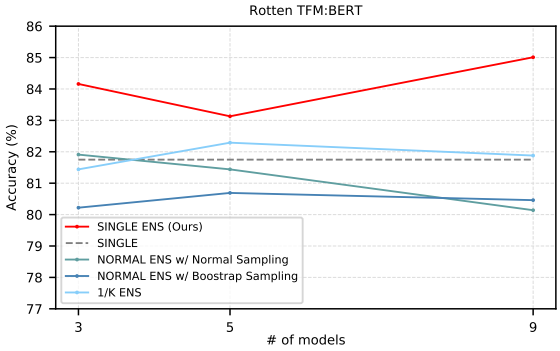

Figure 2: Accuracy depending on the number of models for each ensemble method on the Rotten dataset.

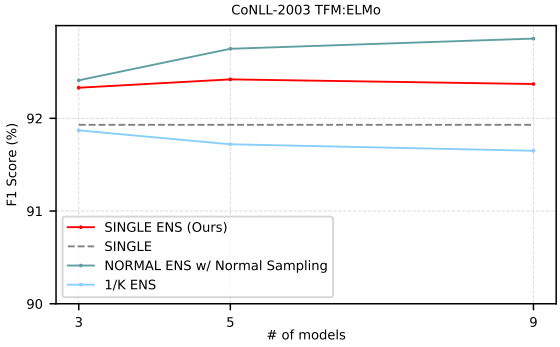

Figure 3: F1 score depending on the number of models for each ensemble method on CoNLL-2003.

Moreover, it has no specific operations to define an explicitly prepared number of virtual models as SINGLEENS has. We conjecture that this difference yields the complementarity that our proposed method and dropout can co-exist.

Vector addition We investigated the patterns with which distinct vectors should be added: 1) Emb, 2) Hidden, and 3) Emb + Hidden. Emb adds distinct vectors only to the embedding, while Hidden adds distinct vectors only to the final feature vectors. Emb + Hidden adds distinct vectors to both the embedding and final feature vectors. As illustrated in Table 4, adding vectors to the embedding is sufficient for improving performance, while adding vectors to hidden vectors has as adverse effect. This observation can be explained by the architecture of Transformer. The distinct vectors in the embedding are recursively propagated through the entire network without being absorbed as non-essential information since the Transformer employs residual connections (He et al., 2015).

Comparison with normal ensembles To evaluate the behavior of our method, we examined the relationship between the performance and the number of models used for training. Our experiments revealed that having more than nine models did not result in significant performance improvement; thus, we only assessed the results up to nine models. Figs 2 and 3 present the metrics on Rotten and CoNLL-2003, respectively. The performance of our method increased with the number of models, which is a general feature of normal ensemble. Notably, on Rotten, the accuracy of our method rose while that of other methods did not. Investigation of this behavior is left for future work.

7 Conclusion

In this paper, we propose a single model ensemble technique called SINGLEENS. The principle of SINGLEENS is to explicitly create multiple virtual models in a single model. Our experiments demonstrated that the proposed method outperformed single models in both text classification and sequence labeling tasks. Moreover, our method with TFM:BERT surpassed the normal ensemble on the IMDB and Rotten datasets, while its parameter size was $1/K$-times smaller. The results thus indicate that explicitly creating virtual models within a single model improves performance. The proposed method is not limited to the two aforementioned tasks, but can be applied to any NLP as well as other tasks such as machine translation and image recognition. Further theoretical analysis can also be performed to elucidate the mechanisms of the proposed method.

Acknowledgment The research results were achieved by "Research and Development of Deep Learning Technology for Advanced Multilingual Speech Translation", the Commissioned Research of National Institute of Information and Communications Technology (NICT), JAPAN. The work was partly supported by JSPS KAKENHI Grant Number 19H04162. We would like to thank Motoki Sato of Preferred Networks and Shun Kiyono of RIKEN for cooperating in preparing the experimental data. We would also like to thank the three anonymous reviewers for their insightful comments.

References

Krogh Anders and Vedelsby Jesper. 1994. Neural network ensembles, cross validation and active learning. In *Proceedings of the 7th International Conference on Neural Information Processing Systems*, (NeurIPS), pages 231–238.

Maas L. Andrew, Daly E. Raymond, Pham T. Peter, Huang Dan, Ng Y. Andrew, and Potts Christopher. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, (ACL), pages 142–150.

Loïc Barrault, Bojar, et al. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation: Shared Task Papers*, (WMT), pages 1–61.

Pang Bo and Lee Lillian. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *CoRR*, pages 115–124.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, (WMT), pages 272–303.

Jacob Devlin, Ming W. Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Monographs on Statistics and Applied Probability. Springer.

Sherif Hashem. 1994. Optimal linear combinations of neural networks. *NEURAL NETWORKS*, 10(4):599–614.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. 2017. Snapshot ensembles: Train 1, get M for free. *CoRR*, abs/1704.00109.

Pennington Jeffrey, Socher Richard, and Manning Christopher. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, (EMNLP), pages 1532–1543.

Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2018. Mixture of expert/imitator networks: Scalable semi-supervised learning framework. *CoRR*, abs/1810.05788.

Hansen K. Lars. and Salamon Peter. 1990. Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 993–1001.

Peters Matthew, Neumann Mark, Iyyer Mohit, Gardner Matt, Clark Christopher, Lee Kenton, and Zettlemoyer Luke. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (NAACL).

Johnson Melvin, Schuster Mike, Le V. Quoc, Krikun Maxim, Wu Yonghui, Chen Zhifeng, Thorat Nikhil, Viégas Fernanda, Wattenberg Martin, Corrado Greg, Hughes Macduff, and Dean Jeffrey. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

David W. Opitz and Jude W. Shavlik. 1996. Actively searching for an effective neural network ensemble. *Connect. Sci.*, 8:337–354.

Sennrich Rico, Haddow Barry, and Birch Alexandra. 2016. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (NAACL), pages 35–40, San Diego, California.

Erik TK. Sang and Fien D. Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning*, (NAACL), pages 142–147.

Erik TK. Sang and Buchholz Sabine. 2000. Introduction to the CoNLL-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

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

Lewis D. Davidand Yang Yiming, Rose G. Tony, and Li Fan. 2004. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. 2020. Incorporating bert into neural machine translation. In *International Conference on Learning Representations*, (ICLR).

A Hyper-parameters and Ensemble Strategy

	Text Classification		Sequence Labeling
	TFM:GLOVE	TFM:BERT	TFM:ELMO
Embedding dimension	200	768	256
Hidden dimension	200	768	256
Number of layers	6	6	6
Number of attention heads	8	8	8
Frozen vectors	GloVe 200	BERT-Large	ELMo 1024
Dropout	-	-	0.5(Emb)
	0.2 (Residual)	0.5 (Residual)	0.2 (Residual)
	0.1 (Attention)	-	0.1 (Attention)
	-	-	0.1 (FF)
Label smoothing	0.1	0.1	-
Optimizer	Adam	Adam	Adam
Initial learning rate	0.0001	0.0001	0.0001
Batch size	64	128	32
Gradient Clipping	1.0	1.0	5.0
Aggrgation Strategy	Averaging	Averaging	Voting
Sampling strategy	Normal & Bootstrapping	Normal & Bootstrapping	Normal

Table 5: Hyper-parameters and ensemble strategies for SINGLE, NORMALENS and SINGLEENS. For TFM:BERT, we followed the model architecture of Zhu et al. (2020). For TFM:ELMO on sequence labeling, we referenced the architecture of Matthew et al. (2018) with replacing the encoder with Transformer. It should be noted that for TFM:ELMO, we add Linear $\rightarrow$ Relu $\rightarrow$ LayerNorm between embedding and self-attention.

	Text Classification		Sequence Labeling
	TFM:GLOVE	TFM:BERT	TFM:ELMO
Embedding dimension	50	64	370
Hidden dimension	50	64	128
Frozen vectors	GloVe 50	BERT-Base	ELMo 256
Number of layers	3	3	4
Number of attention heads	10	8	8
Feed forward dimension	128	128	128
Aggregation Strategy	Averaging	Averaging	Voting
Sampling strategy	Normal	Normal	Normal

Table 6: Hyper-parameters and ensemble strategies for 1/K ENS. The other values are same as Table 5. It should be noted that we ensemble K models of each sub model for final prediction.

B Data Statistics

Task	Dataset	Train	Valid	Test
Text Classification	IMDB	21,246	3,754	25,000
	Rotten	8,636	960	1,066
	RCV 1	14,007	1,557	49,838
Sequence Labeling	CoNLL-2003	14,987	3,466	3,684
	CoNLL-2000	8,926	2,012	2,012

Table 7: Summary of the datasets. The values are the number of sentences contained in each dataset.