Fighting Randomness with Randomness: Mitigating Optimisation Instability of Fine-Tuning using Delayed Ensemble and Noisy Interpolation

Branislav Pecher[♠†], **Jan Cegin**[♠†], **Robert Belanec**[♠†],
Jakub Simko[†], **Ivan Srba**[†], **Maria Bielikova**[†]
[♠] Faculty of Information Technology, Brno University of Technology, Brno, Czechia
[†] Kempelen Institute of Intelligent Technologies, Bratislava, Slovakia
{name.surname}@kinit.sk

Abstract

While fine-tuning of pre-trained language models generally helps to overcome the lack of labelled training samples, it also displays model performance instability. This instability mainly originates from randomness in initialisation or data shuffling. To address this, researchers either modify the training process or augment the available samples, which typically results in increased computational costs. We propose a new mitigation strategy, called **Delayed Ensemble with Noisy Interpolation** (DENI), that leverages the strengths of ensembling, noise regularisation and model interpolation, while retaining computational efficiency. We compare DENI with 9 representative mitigation strategies across 3 models, 4 tuning strategies and 7 text classification datasets. We show that: 1) DENI outperforms the best performing mitigation strategy (Ensemble), while using only a fraction of its cost; 2) the mitigation strategies are beneficial for parameter-efficient fine-tuning (PEFT) methods, outperforming full fine-tuning in specific cases; and 3) combining DENI with data augmentation often leads to even more effective instability mitigation.

1 Introduction

Tuning of pre-trained language models such as BERT or RoBERTa using either full fine-tuning or parameter efficient fine-tuning (PEFT) has achieved significant success across a wide range of natural language processing tasks. They are especially useful when faced with limited labelled data for quickly adapting to the specific task. Despite the success, previous works observed that fine-tuning still remains unstable (Dodge et al., 2020; Mosbach et al., 2021; Chen et al., 2022), especially with limited data. Fine-tuning is sensitive to the effects of randomness originating from random initialisation, data shuffling or model randomness (e.g., use of non-deterministic layers such as dropout in the model). As illustrated in Figure 1,

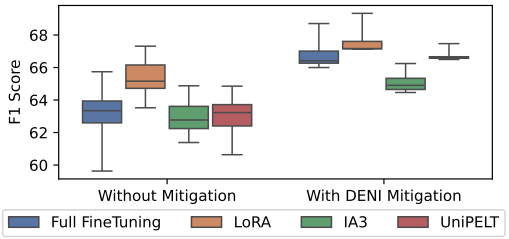

Figure 1: Repeating BERT fine-tuning multiple times without any mitigation leads to significant performance variance. Using DENI for randomness mitigation, the variance is reduced and performance increased.

repeating the fine-tuning process multiple times, without mitigating the randomness, leads to large performance variance in the results, both for full fine-tuning and for PEFT methods.

To deal with the fine-tuning instability, researchers propose various strategies to mitigate the effects of randomness (Pecher et al., 2024). In most cases, the mitigation strategies focus on modifying the training process (Lee et al., 2020; Dodge et al., 2020), such as adding noise (Hua et al., 2021; Wu et al., 2022; Hua et al., 2023), ensembling multiple models (Hidey et al., 2022; Khurana et al., 2021; Wang et al., 2023; Summers and Dinneen, 2021), or improving the experimental setup that potentially leads to the instability (Zhang et al., 2021; Mosbach et al., 2021), such as using bias correction or training for longer. As many of these strategies are designed and evaluated on high-resource datasets, almost no focus is dedicated to evaluating the mitigation benefit of data augmentation (Zhang et al., 2018; Meng et al., 2023a). At the same time, the instability of the whole training process is addressed only when considering full fine-tuning. When using PEFT methods, the focus is on the initialisation of soft prompts, even though the factors such as data shuffling still lead to variance in results (Chen et al., 2022). Overall, the best performing mitigation strategies are ensembles and model interpola-

Findings of the Association for Computational Linguistics: EACL 2024, pages 11005–11044

tion methods (Gueta et al., 2023; Hidey et al., 2022; Wang et al., 2023), which significantly reduce the deviation in results, but also significantly increase the computation costs. The methods that add noise to the model parameters (Hua et al., 2021; Wu et al., 2022) also perform well, improving generalisability and overall performance, but not necessarily reducing the instability.

Inspired by the success and mutual complementarity between ensemble methods, model interpolation and noise regularisation, we propose a novel mitigation method **Delayed Ensemble with Noisy Interpolation** (DENI). The **DENI** method leverages the benefits of ensembling while reducing its computation costs. To achieve this, the ensemble is created at the end of training from a single model by perturbing its parameters using random noise (i.e., fighting randomness with randomness). In addition, the method creates the ensemble by adding noise, trains it for a few steps and then aggregates it into a single model multiple times during training (see Figure 2), which leads to more effective mitigation of the randomness. Using the DENI method leads to lower variance in results and higher performance (as illustrated in Figure 1).

To evaluate the benefit of the DENI method, we compare it with other representative mitigation strategies that modify the optimisation process. In the comparison, we also include an augmentation strategy that uses large language models to paraphrase samples, as such paraphrasing was observed to improve robustness and stability by Cegin et al. (2023, 2024), especially in limited data settings. Besides full fine-tuning, we also explore the benefit of the proposed mitigation strategy and other baselines for representative parameter-efficient fine-tuning (PEFT) methods, namely LoRA, IA3 and UniPELT.

Our main contributions and findings are[1]:

- We propose DENI - a novel strategy for mitigating the randomness sensitivity of fine-tuning (originating from initialisation, data shuffling and model randomness). The proposed method leverages the benefit of ensembling and model interpolation, while reducing their computation costs using noise.
- We compare DENI with 9 representative mitigation strategies, which either modify the training process or augment the data, across 7 text classification datasets. The results show that, in comparison with the best performing baselines, the DENI method improves the overall performance and reduces the deviation in results, while introducing lower computation costs, leading to an efficient mitigation.
- We explore the benefit of mitigation strategies across 3 representative PEFT methods. We find they often benefit more from the mitigation strategies, especially data augmentation, showing a larger decrease in deviation and increase in overall performance, even outperforming full fine-tuning in specific cases.

[1]To support replicability and extension of our results, we openly publish the source code of our method and experiments at https://github.com/kinit-sk/DENI

2 Related Work: Mitigating Instability of Fine-Tuning

The majority of strategies for mitigating the instability of fine-tuning modify the training process or the models themselves, addressing different sources of instability. One set of strategies addresses the suboptimal setup choices, suggesting that training for longer, using an optimiser with bias correction and lower learning rate with warm-up and scheduling can reduce the instability (Mosbach et al., 2021; Zhang et al., 2021). Another set of approaches modifies the initialisation, such as using meta-learning (Dauphin and Schoenholz, 2019), re-initialising top layers of the pretrained models (Zhang et al., 2021), or initialising multiple models and stopping the ones that show bad performance early in training (Dodge et al., 2020). Other strategies optimise only parts of the network (Xu et al., 2021; Zhang et al., 2022) or run supplementary pre-training on data rich tasks (Phang et al., 2018). Mixout (Lee et al., 2020) randomly replaces parts of the parameters with the original weights. Further strategies add noise to the input or model parameters either before (Wu et al., 2022) or during (Hua et al., 2021, 2023; Chen et al., 2023) training. The ensembling strategies provide the best mitigation effectiveness, i.e., highest increase in performance and reduction in results variance, but at the cost of significant increase in computation costs (Hidey et al., 2022; Wang et al., 2023; Gueta et al., 2023). To reduce the cost, the ensemble can be created from a single model, such using Stochastic Weight Averaging (SWA) that performs an equal average of the weights traversed by the optimiser (Izmailov et al., 2018; Khurana et al., 2021; Lu et al., 2022), or Accelerated En-

sembling that uses snapshots from different parts of training (Summers and Dinneen, 2021; Huang et al., 2017). Another possibility is to share the lower layers and ensemble only the classification heads (Chang et al., 2023; Liang et al., 2022). However, such aggregation also reduces the mitigation effectiveness of the ensemble as it reduces the diversity of the models.

Although many augmentation strategies exist, they are mainly used to improve the overall performance in low-resource settings, without any consideration for the instability (Feng et al., 2021; Obadinma et al., 2023). Only a few papers consider using data augmentation for addressing the optimisation instability by utilising pretrained language models as data generators (Cegin et al., 2024; Meng et al., 2023b) or using common augmentation methods such as Mixup (Guo et al., 2019; Guo, 2020).

For the parameter-efficient fine-tuning (PEFT) methods, the focus is mostly on initialising prompts and mitigating randomness in prompt-tuning (Chen et al., 2023; Razdaibiedina et al., 2023; Pan et al., 2023; Köksal et al., 2023), while the focus on mitigating optimisation instability for the remaining PEFT methods is limited (Chen et al., 2022).

3 Mitigation Method: Delayed Ensemble With Noisy Interpolation

The goal of our proposed method, the *Delayed Ensemble with Noisy Interpolation* (DENI), is to mitigate the randomness in fine-tuning optimisation, reducing the observed variability in results, while keeping or increasing the average model performance and minimising additional computational costs. When designing our method, we drew inspiration from findings and approaches from previous works on ensembling (Summers and Dinneen, 2021), model interpolation (Gueta et al., 2023) and noise regularisation (Hua et al., 2021; Wu et al., 2022; Chen et al., 2023), that were shown to be effective techniques capable of improving overall performance and mitigating randomness. Our method, which is illustrated in Figure 2 and in Algorithm 1, comprises two main components: 1) **Delayed Ensemble**; and 2) **Noisy Interpolation**.

Delayed Ensemble (DE). The main idea is to exploit the benefit of ensembling multiple models, while reducing the computation cost of obtaining such an ensemble. It was previously shown, that independently training multiple models (initialised with different random seeds) and aggregating their predictions represents an effective regularisation technique capable of reducing the deviation in the results and significantly improving performance. In addition, Gueta et al. (2023) show that linear interpolation of multiple models in the weight space leads to a better performing model than any of the individual ones used for its aggregation. However, training multiple models with different set of weights introduces a significant increase in computation costs. To reduce the computation cost, while keeping as many of the benefits as possible, we propose the **Delayed Ensemble** that creates the ensemble by adding noise to a single trained model. Instead of initialising multiple models and training them to full, we initialise and train only a single model M. The model is trained for $ensemble_{start}$ steps (E in Figure 2), and afterwards is used to create N new models $(M_1, M_2, ..., M_N)$ by perturbing its parameters by adding noise as follows:

$$M_n = M + Noise(0, var_{noise}) * \lambda_{ensemble} \quad (1)$$

where $Noise(0, var)$ represents Gaussian noise with 0 mean and variance of size var, and $\lambda_{ensemble}$ represents a noise scaling factor for the ensemble. The models are trained for the remainder of the steps, and the prediction is obtained using hard voting (point F in Figure 2).

Algorithm 1 Delayed Ensemble with Noisy Interpolation

Require: var_{noise}: noise variance
$noise_{start}$: steps before adding noise
$noise_{end}$: steps after which no noise is added
$ensemble_{start}$: steps before creating final ensemble
$steps_{noisy}$: number of steps with noisy models
$steps_{regular}$: number of steps with single aggregated model
λ: scaling factor for noise
M: pretrained model
N: number of models in ensemble

Train model M for $noise_{start}$ steps
Set $steps = noise_{start}$
while $steps < noise_{end}$ **do**
  for all n in $1, 2, ..., N$ **do**
    $M_n = M + Noise(0, var_{noise}) * \lambda_{steps}$
  end for
  Train $M, M_1, ..., M_N$ for $steps_{noisy}$
  $M = average(M, M_1, ..., M_N)$
  Train M for $steps_{regular}$
  $steps = steps + steps_{noisy} + steps_{regular}$
end while
Train M for $ensemble_{start} - noise_{end}$
for all n in $1, 2, ..., N$ **do**
  $M_n = M + Noise(0, var_{noise}) * \lambda_{ensemble}$
end for
Train $M, M_1, ..., M_N$ for the remaining steps
Use hard voting of the final ensemble $M, M_1, ..., M_N$

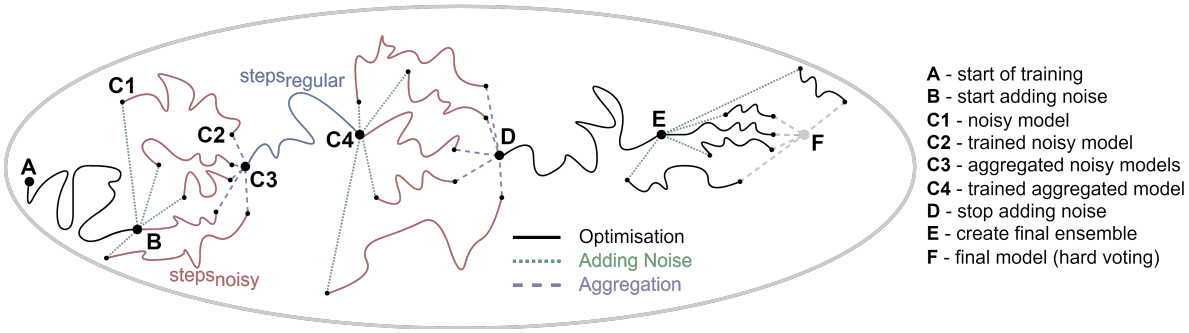

Figure 2: An illustrative example of the Delayed Ensemble with Noisy Interpolation (DENI) method that mitigates model performance instability of model fine-tuning with limited data. DENI alters regular fine-tuning, using noise-adding, model aggregation, and ensembling to steer model(s) towards optimal parameter setup in the parameter space. In comparison to simple ensembling, the method requires only a fraction of computational resources.

Noisy Interpolation (NI). Besides allowing us to create the ensemble from a single trained model, adding noise to the model before, during or even after training was found to be an effective regularisation by itself that can improve the overall performance, generalisability and mitigate the effects of randomness in training (Hua et al., 2021; Wu et al., 2022; Hua et al., 2023). At the same time, training the model that is a result of the linear interpolation of multiple models is more effective and leads to better performance than the further training of the individual models used for aggregation, especially when the interpolated model is in close proximity to the optimal set of parameters (Gueta et al., 2023). Based on these findings, we propose a repeated **Noisy Interpolation**, which creates and interpolates the ensemble multiple times during training. After initialising the model M (A in Figure 2), we train it for $noise_{start}$ steps (B in Figure 2). Afterwards, we create N new models $(M_1, M_2, ..., M_N)$ by perturbing the parameters of model M by adding noise (C1 in Figure 2), similarly to *Delayed Ensemble*. However, we use a different scaling factor, λ_{steps}, which performs inverse scaling based on the number of steps (i.e., in later stages of the training, the introduced noise is smaller). These noisy models are then trained for $steps_{noisy}$ steps (ending in C2 in Figure 2) and are aggregated together using uniform interpolation (C3 in Figure 2). The aggregated model is then trained for $steps_{regular}$ steps (ending in C4 in Figure 2). The whole process of creating and interpolating ensembles is repeated until $noise_{end}$ steps of training are finished (D in Figure 2). Afterwards, the aggregated model is trained until the end.

Full Method (DENI) is the combination of the two components, i.e., running *Noisy Interpolation* for large part of the training before creating final *Delayed Ensemble* near the end.

Hyperparameters. As we define multiple hyperparameters for the proposed method that can affect its mitigation effectiveness and efficiency, we provide recommendations on how to set them up for optimal performance. For the most important parameters (that represent the trade-off between mitigation and computations costs) we run a thorough hyperparameter sensitivity analysis in Appendix B. For the remaining hyperparameters we provide suggestions based on heuristics and our preliminary experiments. Although there are multiple possibilities for the $Noise()$ function, we have observed that using the Gaussian noise provides the best results. In addition, we suggest to add noise only to the newly initialised parameters. Adding noise to all layers makes the method significantly more sensitive to the other hyperparameters (especially var_{noise} and $steps_{noisy}$) – we observed that finding the optimal setup in such case becomes complicated, as adding slightly larger noise completely breaks the model (i.e., performing worse than random), while adding slightly smaller noise leads to negligible mitigation effects. Finally, similar to Wu et al. (2022), we found that scaling the noise based on the standard deviation of the model parameters provides the most benefit (again reducing hyperparameter sensitivity). At the same time, scaling the noise based on the number of steps is beneficial for the *Noisy Interpolation*, but not *Delayed Ensemble*. As such, we define the scaling factors in the following way: $\lambda_{steps} = std(W_i) * \frac{1}{steps}$ and $\lambda_{ensemble} = std(W_i)$, where $std(Wi)$ represents

the standard deviation of the parameter to which we are adding the noise. Overall, the DENI method is not as sensitive to the hyperparameter setup, even though the number of parameters is larger. In majority of the cases, the optimal parameters can be determined based on heuristics, while slight change leads to only a small difference in mitigation effectiveness, but only if using scaled noise and only for the newly initialised model parameters.

Reducing Memory Requirements and Inference Cost. Although we mainly focus on the computation cost of training the models for the ensemble, the proposed strategy has a positive impact on memory requirements and the inference cost as well. As the noise is added only to the newly initialised parameters and the models in ensemble are created from a single model, majority of the parameters can be shared between the models in the ensemble. Thanks to the weight sharing (similar to Chang et al. (2023); Liang et al. (2022)), the memory requirements and inference cost can be reduced significantly, being only slightly higher than using a single model (only the newly initialised parameters need to be duplicated, while reusing the single forward pass through the majority of the model as the newly initialised parameters are last layers in the network). However, the weight sharing strongly benefit depends on which parameters are perturbed – if adding noise to all parameters or to parameters that are at the start of the model, the possibilities for weight sharing, and its benefits, would be limited.

4 Experimental Results and Findings

Baselines. First, we use two basic baselines without any mitigation: 1) **Default**, where the model is fine-tuned only on the limited data (representing the lower bound of performance); and 2) **All Data**, where the model is fine-tuned on all the samples available in the dataset (representing the upper bound of performance). Second, we compare with other representative approaches that introduce a change into the optimisation process, specifically: 1) **Best Practices** as reported by Mosbach et al. (2021) and Zhang et al. (2021), which includes using scheduling, warm-up, optimiser with bias correction and training for longer; 2) **Ensemble**, where we concurrently train 10 models with different initialisation and use hard voting to aggregate their predictions; 3) **Noise**$_{Input}$, where we periodically add noise to the input embeddings during training; 4) **Noise**$_{Weights}$, where we periodically add noise to the parameters during training; 5) **Stochastic Weight Averaging (SWA)** (Izmailov et al., 2018; Khurana et al., 2021; Lu et al., 2022) that performs an equal average of the weights traversed by the optimiser with modified learning rate schedule; and 6) **Mixout** (Lee et al., 2020) that stochastically mixes the parameters of two models instead of dropout. In addition, we also compare with **Augment N** (Cegin et al., 2023, 2024), which uses a large language model to paraphrase each sample N times to extend the training data (we report results only for the best performing number of paraphrases, Augment 1 and 2). Finally, we compare our proposed method (**Our**$_{DENI}$), its individual components (**Our**$_{DE}$ and **Our**$_{NI}$) and its extended version **Delayed Ensemble with Noisy Interpolation and Augmented Labelled Samples (Our**$_{DENIALS}$) that combines DENI with Augment 1. More details for each of the strategies are reported in Appendix C.2.

Datasets. The experiments are conducted on 7 text classification datasets composed of different tasks with different number of classes. We focus on 3 binary classification datasets from the GLUE benchmark (Wang et al., 2018): **SST2** (Socher et al., 2013) for sentiment classification, **CoLA** (Warstadt et al., 2019) for determining the grammatical acceptability of a sentence, and **MRPC** (Dolan and Brockett, 2005) for determining the semantic equivalence relationship between two sentences. In addition, we use 4 multi-class text datasets: **AG News** (Zhang et al., 2015) for news classification, **TREC** (Voorhees and Tice, 2000) for question classification, **DBPedia** (Lehmann et al., 2015) for topic classification and **SNIPS** (Coucke et al., 2018) for intent classification.

Experimental Setup. As there is no consensus what constitutes a low-resource setting (e.g., for some it is having less than 1000 labelled samples (Chen et al., 2022; Zhang et al., 2021; Gueta et al., 2023; Chen et al., 2023) and for others less than 10 000 (Hua et al., 2021)), we opted to use only 1000 labelled samples from each dataset for training for the main set of experiments and explore different sizes of available training data in an experiment presented in Section 4.3. Each experiment is repeated 20 times with different random seed that affects the initialisation, order of samples and model randomness. For each experiment, we report a mean F1 macro score and standard deviation over these repeated runs. Further experimental setup

details are reported in Appendix C.1.

Models. Each experiment is run using the BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and ALBERT (Lan et al., 2020) base models. Besides regular fine-tuning, we also report results for representative parameter-efficient fine-tuning methods, specifically LoRA (Hu et al., 2022), IA3 (Liu et al., 2022) and UniPELT (Mao et al., 2022) (which combines LoRA, Pfeiffer Adapter (Pfeiffer et al., 2020) and Prefix-Tuning (Li and Liang, 2021)). Further details regarding hyper-parameter setup are reported in Appendix C.1.

4.1 DENI Method on Full Fine-Tuning

In this section, our goal is to answer the following research question: ***RQ1:*** *How does the DENI method perform in comparison to other mitigation strategies?* We compare the proposed method with existing mitigation strategies in terms of the overall performance, standard deviation and their computation cost. The computation cost is calculated using the number of training steps normalised by the steps of the *Default* strategy (i.e., $cost = \frac{n\_steps_{strategy}}{n\_steps_{Default}}$). In addition, we explore what benefit the different components of the DENI method bring. The results from the comparison on full fine-tuning are presented for the BERT model in Table 1, with the full results in Appendix F and visualisation for the relation between performance, deviation and the cost in Appendix D.

Strategies that significantly increase number of steps provide the most benefit. The *Ensemble* represents the most effective mitigation strategy, showing the highest increase in the overall performance (e.g., from 75.11 to 76.54 on CoLA dataset) and reduction in standard deviation (e.g., from 0.662 to 0.325 on CoLA dataset), with the deviation being often lower than the *All Data* baseline. However, it also represent the most computationally expensive strategy (when not considering training on all data where the cost is dataset dependent). The *Ensemble* strategy is closely followed by *SWA*, *Best Practices* and *Augment N* that are less consistent and show lower benefit, but also require lower number of steps. **The *Augment* strategy is beneficial only when using at most 2 paraphrases per samples**, with higher number of paraphrases leading to significant decrease in performance (see Appendix E for results for higher number of N). The remaining mitigation strategies show only negligible and dataset-dependent benefit.

The *DENI* method leads to higher performance and lower standard deviation while requiring lower number of steps than the *Ensemble* strategy. In comparison to the best performing mitigation strategy (*Ensemble*), we observe a statistically significant increase of $0.25 - 2.11$ percentage points in performance (p-value of 7e-5 using Mann-Whitney U test), with the deviation staying similar or even lower in specific cases. At the same time, the *DENI* method requires only 37% of the training steps. As such, it represents an effective and efficient mitigation strategy.

The *DENI* method components provide different benefits and complement each other. The *Delayed Ensemble* component leads to reduction of standard deviation, achieving deviation on par or lower than *Ensemble*, but may not necessarily lead to increase in performance. For example on the MRPC dataset, even though the deviation is lower (0.594 as opposed to 0.674), the performance is also significantly lower (63.68 as opposed to 65.69). However, the performance is higher than the one from *Default* baseline in all cases. On the other hand, the *Noisy Interpolation* leads to significant increase in performance, at the cost of higher deviation in results. For example on the AG News dataset, the performance is increased to 87.67 as opposed to 85.56 from *Ensemble* strategy, with the deviation also increasing to 0.760 as opposed to 0.304. In the full method, the components complement each other, suppressing the weaknesses and amplifying the strengths (e.g., reducing the deviation of NI also leads to higher performance). Finally, combining our proposed method with data augmentation may lead to further benefits, although the overall benefit is lower ($0.2 - 0.6$) and not statistically significant (p-value of 0.14 using Mann-Whitney U test) with a significant cost increase (double the training steps). As such, **combining mitigation strategies that target optimisation with augmentation strategies may lead to further increase** for all strategies.

The effectiveness of the *DENI* method and the different mitigation strategies is consistent across different models. On the RoBERTa and ALBERT model, we still observe similar findings for the *DENI* method and the majority of mitigation strategies, even though the effect of mitigation in absolute number may be different. In addition, the benefit of specific mitigation strategies grows on different models (*SWA* or *Augment* for ALBERT).

BERT	AG NEWS	TREC	SNIPS	DBPEDIA	SST2	MRPC	COLA	COST
				FULL FINETUNING				
DEFAULT	$84.95_{0.482}$	$90.00_{0.682}$	$97.99_{0.109}$	$98.75_{0.047}$	$88.27_{0.230}$	$62.73_{1.497}$	$75.11_{0.662}$	1
ALL DATA	$88.65_{0.343}$	$95.66_{0.579}$	$98.86_{0.066}$	$99.21_{0.030}$	$95.15_{0.078}$	$68.86_{0.893}$	$79.56_{0.440}$	$5-55$
BEST PRACTICES	$84.96_{0.444}$	$90.40_{0.585}$	$98.05_{0.137}$	$98.76_{0.049}$	$88.32_{0.313}$	$63.07_{1.593}$	$75.30_{0.718}$	2
ENSEMBLE	$85.56_{0.304}$	$90.67_{0.400}$	$98.22_{0.076}$	$98.82_{0.021}$	$\underline{89.05_{0.079}}$	$65.69_{0.674}$	$76.54_{0.325}$	10
NOISE$_{Input}$	$85.07_{0.444}$	$89.43_{0.775}$	$98.09_{0.114}$	$98.77_{0.043}$	$88.25_{0.285}$	$62.44_{1.499}$	$75.05_{0.817}$	1
NOISE$_{Weights}$	$85.25_{0.671}$	$90.20_{0.860}$	$98.03_{0.146}$	$98.75_{0.061}$	$88.04_{0.266}$	$62.68_{1.765}$	$75.06_{0.699}$	1
SWA	$85.40_{0.364}$	$90.24_{0.632}$	$98.06_{0.122}$	$98.76_{0.060}$	$88.54_{0.211}$	$63.50_{1.751}$	$75.26_{0.572}$	1.75
MIXOUT	$84.96_{0.538}$	$90.20_{0.603}$	$98.03_{0.111}$	$98.73_{0.040}$	$88.39_{0.280}$	$63.23_{1.137}$	$75.33_{0.572}$	1
AUGMENT 1	$85.25_{0.542}$	$90.36_{0.945}$	$98.19_{0.142}$	$98.82_{0.082}$	$88.66_{0.286}$	$63.01_{1.544}$	$75.56_{0.443}$	2.4
AUGMENT 2	$85.28_{0.433}$	$90.45_{0.676}$	$98.10_{0.143}$	$98.72_{0.076}$	$88.63_{0.327}$	$62.40_{1.514}$	$74.58_{0.426}$	3.6
OUR$_{DE}$	$85.17_{0.291}$	$90.54_{0.278}$	$98.10_{0.086}$	$\underline{98.76_{0.019}}$	$88.43_{0.107}$	$63.68_{0.594}$	$75.09_{0.281}$	1.9
OUR$_{NI}$	$86.81_{0.760}$	$91.42_{0.901}$	$98.42_{0.131}$	$98.99_{0.044}$	$90.08_{0.284}$	$65.38_{1.571}$	$77.16_{0.586}$	2.8
OUR$_{DENI}$	$87.67_{0.261}$	$92.04_{0.377}$	$98.65_{0.099}$	$\mathbf{99.07}_{0.023}$	$90.82_{0.129}$	$66.66_{0.613}$	$77.74_{0.263}$	3.7
OUR$_{DENIALS}$	$\underline{\mathbf{88.18}_{0.196}}$	$\underline{\mathbf{92.27}_{0.236}}$	$\underline{\mathbf{98.73}_{0.075}}$	$99.03_{0.040}$	$\mathbf{91.22}_{0.132}$	$\underline{\mathbf{67.04}_{0.544}}$	$\underline{\mathbf{78.29}_{0.153}}$	7.4

Table 1: Comparison of the DENI method with existing mitigation strategies and baselines on full fine-tuning using BERT. The comparison is done in terms of overall performance, deviation and cost (normalised training steps). The highest performance is in **bold** and lowest deviation is underlined (not considering *All Data* baseline).

BERT	AG NEWS	TREC	SNIPS	DBPEDIA	SST2	MRPC	COLA
				LORA			
DEFAULT	$85.10_{0.488}$	$89.68_{0.770}$	$98.06_{0.142}$	$98.72_{0.077}$	$87.95_{0.315}$	$61.70_{6.377}$	$74.80_{0.907}$
ENSEMBLE	$86.32_{0.266}$	$91.58_{0.371}$	$98.32_{0.125}$	$98.86_{0.030}$	$89.17_{0.109}$	$66.59_{0.753}$	$77.02_{0.533}$
OUR$_{DENI}$	$87.86_{0.305}$	$92.39_{0.377}$	$98.67_{0.084}$	$99.05_{0.041}$	$90.74_{0.122}$	$67.49_{0.554}$	$77.66_{0.425}$
OUR$_{DENIALS}$	$87.79_{0.244}$	$92.24_{0.204}$	$98.65_{0.065}$	$99.04_{0.038}$	$91.03_{0.206}$	$66.79_{0.671}$	$78.24_{0.041}$
				IA3			
DEFAULT	$83.83_{0.829}$	$88.42_{1.197}$	$97.59_{0.259}$	$98.56_{0.105}$	$87.60_{0.216}$	$62.72_{0.979}$	$73.56_{0.963}$
ENSEMBLE	$85.39_{0.327}$	$90.54_{0.304}$	$98.16_{0.082}$	$98.78_{0.034}$	$88.55_{0.092}$	$64.54_{0.767}$	$75.99_{0.440}$
OUR$_{DENI}$	$86.74_{0.279}$	$90.84_{0.238}$	$98.40_{0.117}$	$99.03_{0.041}$	$90.15_{0.121}$	$65.00_{0.473}$	$76.70_{0.295}$
OUR$_{DENIALS}$	$87.08_{0.574}$	$92.26_{0.127}$	$98.42_{0.128}$	$99.04_{0.042}$	$90.35_{0.224}$	$67.02_{0.378}$	$78.30_{0.037}$
				UNIPELT			
DEFAULT	$84.97_{0.582}$	$87.13_{1.316}$	$97.63_{0.194}$	$98.56_{0.262}$	$87.82_{0.233}$	$63.77_{0.798}$	$74.91_{0.403}$
ENSEMBLE	$86.02_{0.291}$	$91.08_{0.559}$	$98.31_{0.104}$	$98.89_{0.025}$	$88.63_{0.141}$	$64.07_{0.610}$	$76.00_{0.458}$
OUR$_{DENI}$	$87.54_{0.301}$	$91.43_{0.678}$	$98.56_{0.084}$	$99.06_{0.045}$	$90.57_{0.078}$	$66.72_{0.275}$	$77.68_{0.409}$
OUR$_{DENIALS}$	$87.44_{0.054}$	$92.25_{0.088}$	$98.56_{0.068}$	$99.04_{0.045}$	$91.01_{0.157}$	$66.59_{0.323}$	$78.24_{0.033}$

Table 2: Comparison of the best performing mitigation strategy, *DENI* method and the *Default* baseline for different parameter-efficient fine-tuning methods. The comparison is done in terms of overall performance and deviation.

4.2 Mitigation Strategies for PEFT Methods

In this section, we aim to answer the following research question: ***RQ2:*** *What is the benefit of mitigation strategies for parameter-efficient fine-tuning (PEFT) methods?* We compare the mitigation strategies and the *DENI* method across different PEFT methods and observe how the increase in overall performance and the reduction in standard deviation changes. The results for the BERT model on the best performing mitigation strategies is in Table 2, for all strategies on the SNIPS dataset in Figure 3, with the full results in Appendix F.

Combination of the PEFT methods with the mitigation strategies can often lead to more effective mitigation. Compared to full fine-tuning combination of *Ensemble* with LoRA or UniPELT leads to higher increase in performance (86.32 for LoRA and 86.02 for UniPELT compared to 85.56 for full fine-tuning on TREC) and reduction in standard deviation (0.266 and 0.291 compared to 0.304). Similar behaviour can be observed on the *Default* baseline as well, where with lower number of trained parameters, the performance or the standard deviation is on par, or even better, than when training all parameters. For example, we observe deviation of 0.403 on CoLA dataset with UniPELT as compared to 0.662 for full fine-tuning. As such, when faced with limited labelled data **the instability of training can be addressed by reducing the number of trainable parameters.**

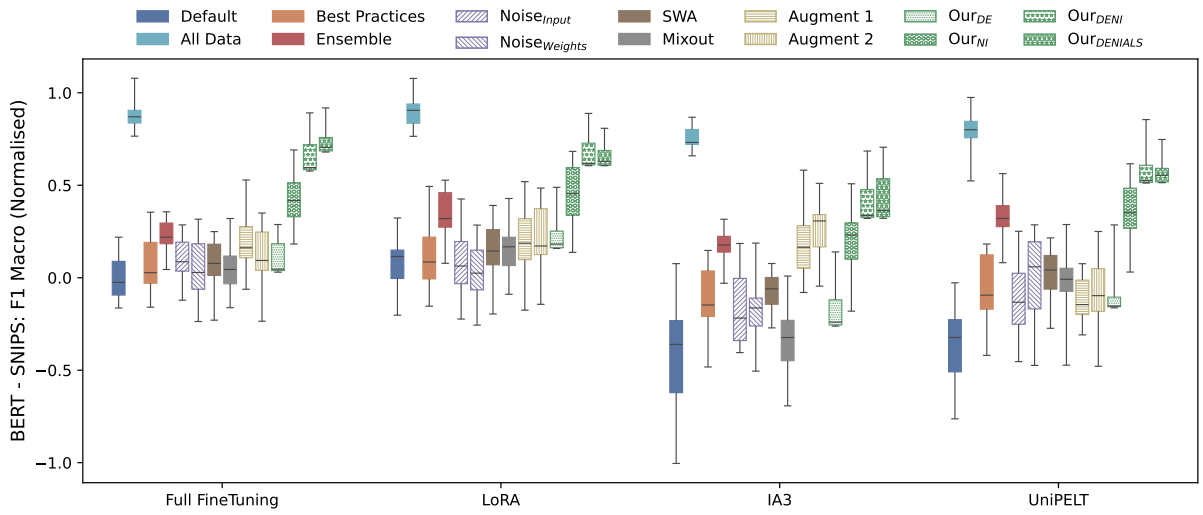

Figure 3: Benefit of mitigation strategies for the different fine-tuning methods using BERT on SNIPS dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3). In addition, the *DENI* method outperforms all mitigation strategies, leading to higher performance and lower deviation.

The *Augment* mitigation strategy provides more benefit when reducing the number of trainable parameters. The IA3 method, which trains approximately 0.5% parameters, benefits significantly more from using augmented data. For example, we observe an increase in performance of up to 2 percentage points on the MRPC dataset for the *DENIALS* method over the *DENI* method. At the same time, the *Augment N* often outperforms the *Ensemble* in terms of performance when using the IA3 method, but not necessarily in terms of deviation. The performance benefit of data augmentation is lower for other PEFT method, specifically up to 0.6 percentage points for LoRA (CoLA dataset) and up to 0.8 percentage points for UniPELT (SST2 dataset). Similarly, using augmented data leads to a significantly larger performance increase for ALBERT and lower increase for RoBERTa. Finally, **using augmented data in combination with the *DENI* method often leads to lower deviation in results across all tuning methods**, especially for PEFT. For example, using UniPELT the deviation drops from 0.301 to 0.054 on AG News dataset, or from 0.678 to 0.088 on TREC dataset. **Similar results can be observed across different models for all of the datasets and tuning methods.**

4.3 Behaviour on Different Sizes of Dataset

Our goal in this section is to answer the following research question: ***RQ3:*** *How does the number of available labelled samples (shots) affect the benefit and effectiveness of mitigation strategies?* We vary the number of available samples from 1 per class up to 1000 per class and compare the *Ensemble* and *DENI* method with the *Default* and *All Data* baselines. The results of this ablation study for the BERT model on the TREC dataset presented in Figure 4 (the number of samples ranging from 6 in 1-shot setting up to 6000 in 1000-shot setting).

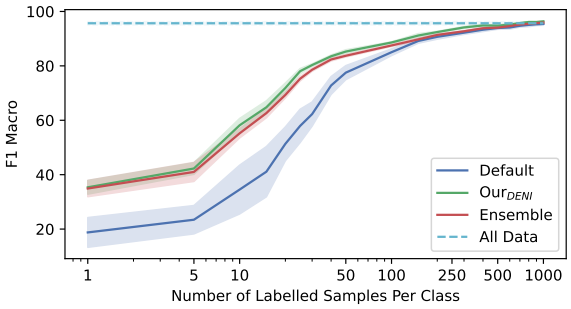

Figure 4: Mitigation effectiveness across different dataset sizes for the BERT model on TREC dataset. Benefit of mitigation strategies is higher on lower number of shots and gradually decrease with more shots.

Mitigation strategies provide larger benefit on low number of samples. The difference in performance between the mitigation strategies and the *Default* baseline is as high as 20 percentage points (80% compared to 60% when using 30 shots), with significantly lower standard deviation (0.453 compared to 4.34). As we increase the number of shots, the benefit gradually decreases. However, **the mitigation strategies provide their benefit even when**

using all data for training, increasing the performance by up to 1 percentage point and reducing deviation by up to 75% (from 0.448 to 0.116).

The *DENI* method outperforms *Ensemble* across all dataset sizes. The performance difference oscillates between $0.4 - 3$ percentage points across the different shots. Similarly, the deviation stays the same, or even lower for the *DENI* method, especially on lower number of shots (e.g., 2.095 compared to 3.418 when using 5 shots).

5 Conclusions

In this work, we have proposed a novel method for effective and efficient mitigation of the instability in fine-tuning of pre-trained language models based on ensembling, model interpolation and noise regularisation. We compare the method with prior mitigation strategies, showing it leads to stronger mitigation, with lower computation cost. In addition, we show that the mitigation strategies provide more benefit to the PEFT methods, such as LoRA, IA3 or UniPELT. Combining mitigation strategies with PEFT can often lead to higher performance and lower deviation than their combination with full fine-tuning. Moreover, we show that data augmentation provides more benefit when limiting the number of trainable parameters, such as using IA3 or training smaller models.

Acknowledgements

This work was partially supported by *VIGILANT*, a project funded by the European Union under the Horizon Europe under the Contract No. 101073921, the *Central European Digital Media Observatory 2.0 (CEDMO 2.0)*, a project funded by the European Union under the Contract No. 101158609, and the *MODERMED*, a project funded by the Slovak Research and Development Agency under GA No. APVV-22-0414.

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254).

Limitations

The setting we focus on in the main experiments may obscure the overall benefit and effectiveness of different mitigation strategies. First, it was previously observed that smaller models with lower number of parameters show lower sensitivity to the effects of randomness when working with limited labelled data. As we use smaller models, namely base versions of BERT, RoBERTa and ALBERT, the benefit of our proposed method as well as other mitigation strategies may be lower than if we used larger models. In addition, we are quite conservative with what we consider as low resource setting and treat availability of 1000 labelled samples to already represent such. However, as we observed in the experiments, the performance is already quite high and the deviation quite low on some datasets even with this number of labelled samples. In addition, in the experiments from Section 4.3, we observe that the benefit of mitigation strategies on significantly lower number of samples per class is significantly higher. As such, this may obscure the real benefit of different mitigation strategies. However, our choices in both cases are based on previous works. As discussed in the Experimental Setup, many previous works consider 1000 labelled samples to already be low resource setting. At the same time, many of the mitigation strategies were designed and evaluated on high-resource setting with full datasets. Finally, we choose the smaller models in order to provide more extensive analysis and comparison, while limiting the impact (in terms of generated CO2) as much as possible.

Even though a large number of mitigation strategies exist, we focus on a smaller set of representative strategies. This is especially evident with augmentation strategies, where we focus only on augmentation using the paraphrasing from pretrained large language models. However, when choosing the strategies for comparison, we specifically focused on selecting the most representative ones, i.e., those that achieved the best mitigation (in terms of performance and standard deviation) in related work (and in some cases in our pilot experiments) from different groups of mitigation strategies. As such, even though we could increase the number strategies we compare to, it would not provide any additional findings. At the same time, our goal is to limit the potential negative impact of our work as much as possible and therefore have focused only on this set of samples.

Finally, we do not consider prompt-based parameter-efficient fine-tuning approaches (P-tuning (Liu et al., 2023), Prefix-tuning (Li and Liang, 2021) or Prompt-tuning (Lester et al., 2021)), even though they are one of the most popular PEFT methods that are currently used for many pretrained large language models. However, we specifically focus on fine-tuning with typical classification models (that use classification head at

the end and do not perform classification through prompting), and as such the prompt-based PEFT methods would not provide any significant benefit as they were specifically designed for training models for prompting and in-context learning. In addition, the UniPELT method already uses Prefix-tuning (which may be the most relevant for typical classification models) as one of the aggregated methods – however, we have observed that this PEFT method performed the worst in our experiments and was especially sensitive to the hyperparameter setup (both for the models and for the mitigation strategies).

References

Jan Cegin, Branislav Pecher, Jakub Simko, Ivan Srba, Maria Bielikova, and Peter Brusilovsky. 2024. Effects of diversity incentives on sample diversity and downstream model performance in llm-based text augmentation. *Preprint*, arXiv:2401.06643.

Jan Cegin, Jakub Simko, and Peter Brusilovsky. 2023. ChatGPT to replace crowdsourcing of paraphrases for intent classification: Higher diversity and comparable model robustness. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1889–1905, Singapore. Association for Computational Linguistics.

Haw-Shiuan Chang, Ruei-Yao Sun, Kathryn Ricci, and Andrew McCallum. 2023. Multi-CLS BERT: An efficient alternative to traditional ensembling. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–854, Toronto, Canada. Association for Computational Linguistics.

Guanzheng Chen, Fangyu Liu, Zaiqiao Meng, and Shangsong Liang. 2022. Revisiting parameter-efficient tuning: Are we really there yet? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2612–2626, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Lichang Chen, Jiuhai Chen, Heng Huang, and Minhao Cheng. 2023. PTP: Boosting stability and performance of prompt tuning with perturbation-based regularizer. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13512–13525, Singapore. Association for Computational Linguistics.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Yann N Dauphin and Samuel Schoenholz. 2019. Metainit: Initializing learning by learning to initialize. *Advances in Neural Information Processing Systems*, 32.

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.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, Online. Association for Computational Linguistics.

Almog Gueta, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. 2023. Knowledge is a region in weight space for fine-tuned language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1350–1370, Singapore. Association for Computational Linguistics.

Hongyu Guo. 2020. Nonlinear mixup: Out-of-manifold data augmentation for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4044–4051.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*.

Christopher Hidey, Fei Liu, and Rahul Goel. 2022. Reducing model churn: Stable re-training of conversational agents. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 14–25, Edinburgh, UK. Association for Computational Linguistics.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Hang Hua, Xingjian Li, Dejing Dou, Cheng-Zhong Xu, and Jiebo Luo. 2023. Improving pretrained language model fine-tuning with noise stability regularization. *IEEE Transactions on Neural Networks and Learning Systems*.

Hang Hua, Xingjian Li, Dejing Dou, Chengzhong Xu, and Jiebo Luo. 2021. Noise stability regularization for improving BERT fine-tuning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3229–3241, Online. Association for Computational Linguistics.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. 2017. Snapshot ensembles: Train 1, get m for free. In *International Conference on Learning Representations*.

P Izmailov, AG Wilson, D Podoprikhin, D Vetrov, and T Garipov. 2018. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 876–885.

Damjan Kalajdzievski. 2023. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*.

Urja Khurana, Eric Nalisnick, and Antske Fokkens. 2021. How emotionally stable is ALBERT? testing robustness with stochastic weight averaging on a sentiment analysis task. In *Proceedings of the 2nd Workshop on Evaluation and Comparison of NLP Systems*, pages 16–31, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Abdullatif Köksal, Timo Schick, and Hinrich Schuetze. 2023. MEAL: Stable and active learning for few-shot prompting. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 506–517, Singapore. Association for Computational Linguistics.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*.

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. In *International Conference on Learning Representations*.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2020. Mixout: Effective regularization to finetune large-scale pretrained language models. In *International Conference on Learning Representations*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia – a large-scale, multilingual knowledge base extracted from wikipedia. *Semant. Web*, 6(2):167–195.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Chen Liang, Pengcheng He, Yelong Shen, Weizhu Chen, and Tuo Zhao. 2022. CAMERO: Consistency regularized ensemble of perturbed language models with weight sharing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7162–7175, Dublin, Ireland. Association for Computational Linguistics.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 1950–1965. Curran Associates, Inc.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023. Gpt understands, too. *AI Open*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Peng Lu, Ivan Kobyzev, Mehdi Rezagholizadeh, Ahmad Rashid, Ali Ghodsi, and Phillippe Langlais. 2022. Improving generalization of pre-trained language models via stochastic weight averaging. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4948–4954, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. 2022. UniPELT: A unified framework for parameter-efficient language model tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6253–6264, Dublin, Ireland. Association for Computational Linguistics.

R. Thomas McCoy, Junghyun Min, and Tal Linzen. 2020. BERTs of a feather do not generalize together: Large variability in generalization across models with

similar test set performance. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 217–227, Online. Association for Computational Linguistics.

Yu Meng, Martin Michalski, Jiaxin Huang, Yu Zhang, Tarek Abdelzaher, and Jiawei Han. 2023a. Tuning language models as training data generators for augmentation-enhanced few-shot learning. In *International Conference on Machine Learning*, pages 24457–24477. PMLR.

Yu Meng, Martin Michalski, Jiaxin Huang, Yu Zhang, Tarek Abdelzaher, and Jiawei Han. 2023b. Tuning language models as training data generators for augmentation-enhanced few-shot learning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning {bert}: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*.

Stephen Obadinma, Hongyu Guo, and Xiaodan Zhu. 2023. Effectiveness of data augmentation for parameter efficient tuning with limited data. *arXiv preprint arXiv:2303.02577*.

Kaihang Pan, Juncheng Li, Hongye Song, Jun Lin, Xiaozhong Liu, and Siliang Tang. 2023. Self-supervised meta-prompt learning with meta-gradient regularization for few-shot generalization. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1059–1077, Singapore. Association for Computational Linguistics.

Branislav Pecher, Ivan Srba, and Maria Bielikova. 2024. A survey on stability of learning with limited labelled data and its sensitivity to the effects of randomness. *ACM Computing Surveys*. Just Accepted.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Anastasiia Razdaibiedina, Yuning Mao, Madian Khabsa, Mike Lewis, Rui Hou, Jimmy Ba, and Amjad Almahairi. 2023. Residual prompt tuning: improving prompt tuning with residual reparameterization. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6740–6757, Toronto, Canada. Association for Computational Linguistics.

Thibault Sellam, Steve Yadlowsky, Ian Tenney, Jason Wei, Naomi Saphra, Alexander D'Amour, Tal Linzen, Jasmijn Bastings, Iulia Turc, Jacob Eisenstein, Dipanjan Das, and Ellie Pavlick. 2022. The MultiBERTs: BERT Reproductions for Robustness Analysis. In *International Conference on Learning Representations*, page 30.

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.

Cecilia Summers and Michael J Dinneen. 2021. Nondeterminism and instability in neural network optimization. In *International Conference on Machine Learning*, pages 9913–9922. PMLR.

Ellen M. Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, page 200–207, New York, NY, USA. Association for Computing Machinery.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Lijing Wang, Yingya Li, Timothy Miller, Steven Bethard, and Guergana Savova. 2023. Two-stage fine-tuning for improved bias and variance for large pretrained language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15746–15761, Toronto, Canada. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2022. NoisyTune: A little noise can help you finetune pretrained language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 680–685, Dublin, Ireland. Association for Computational Linguistics.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. 2021. Raise a child in large language model: Towards effective and generalizable fine-tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9514–9528, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Haojie Zhang, Ge Li, Jia Li, Zhongjin Zhang, Yuqi Zhu, and Zhi Jin. 2022. Fine-tuning pre-trained language models effectively by optimizing subnetworks adaptively. *Advances in Neural Information Processing Systems*, 35:21442–21454.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2021. Revisiting few-sample {bert} fine-tuning. In *International Conference on Learning Representations*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

A Ethical Considerations and Impact Statement

The experiments in this paper work with publicly available benchmark dataset GLUE (SST2, MRPC and CoLA from the benchmark), and publicly available datasets AG News, TREC, SNIPS and DBPedia, citing the original authors. As we were not able to determine the license for all of the tasks and datasets used, we opted to use them in as limited form as possible, adhering to the terms of use (no annotation of the test set) for the GLUE benchmark dataset and applying it to other datasets as well. As the datasets are commonly used, we do not check them for any identifiable information or offensive content, assuming the publicly available classification benchmark datasets do not contain such content (as it was already removed by the authors of the datasets). We do not work with any personally identifiable information or offensive content and perform no crowdsourcing for further data annotation. In addition, we are not aware of any potential ethical harms or negative societal impacts of our work, apart from the ones related to the advancement of the field of Machine Learning and Learning with Limited Labelled Data (mainly the use of computation resources, consuming energy and generating CO2). Finally, we follow the license terms for all the models we use – all models and datasets allow their use as part of research. As we perform only classification and do not release the predictions, we generate no potentially biased or offensive content.

Impact Statement: CO2 Emissions Related to Experiments The experiments presented in this paper used significant compute resources as we train multiple models (3) over multiple random seeds (20), for different mitigation strategies and baselines (17), fine-tuning methods (4) and datasets (7). Overall, all the experiments (including preliminary experiments for which we do not report results in this paper) were conducted using a private infrastructure, which has a carbon efficiency of 0.432 kgCO_2eq/kWh. A cumulative of approximately 1000 hours of computation was performed on hardware of type A100 PCIe 40GB (TDP of 250W). Total emissions are estimated to be 108 kgCO_2eq of which 0 percent were directly offset. These estimations were conducted using the MachineLearning Impact calculator presented in Lacoste et al. (2019). Whenever possible, we tried to reduce the compute resources used as much as possible. We evaluate on smaller models (base versions instead of large). In addition, we evaluate and compare our method only to a set of representative mitigation strategies, determined based on prior works and our preliminary experiments. Finally, the estimate does not include the cost of generating the paraphrases as the information about the efficiency for the large language model behind API is not available. To limit the impact of the augmentation, we decide to run it only a single time on a set of labelled samples that are used throughout the training.

B Hyperparameter Sensitivity Analysis

In this section, we explore the sensitivity of the *DENI* method to the hyperparameter setup. We focus on the hyperparameters that represents the trade-off between mitigation effectiveness and computation costs. The hypeparameters that directly affect each other are grouped together as follows: 1) the size of noise we are adding (var_{noise}) together with how often the noise is added ($steps_{noisy}$); 2) number of steps after which we start ($noise_{start}$) and stop ($noise_{end}$) adding noise; 3) number of steps after which the final ensemble is created ($ensemble_{start}$); and 4) number of models in the ensemble (N). The results of this analysis are presented in Figure 5 for BERT on TREC dataset.

The number of models in ensemble shows

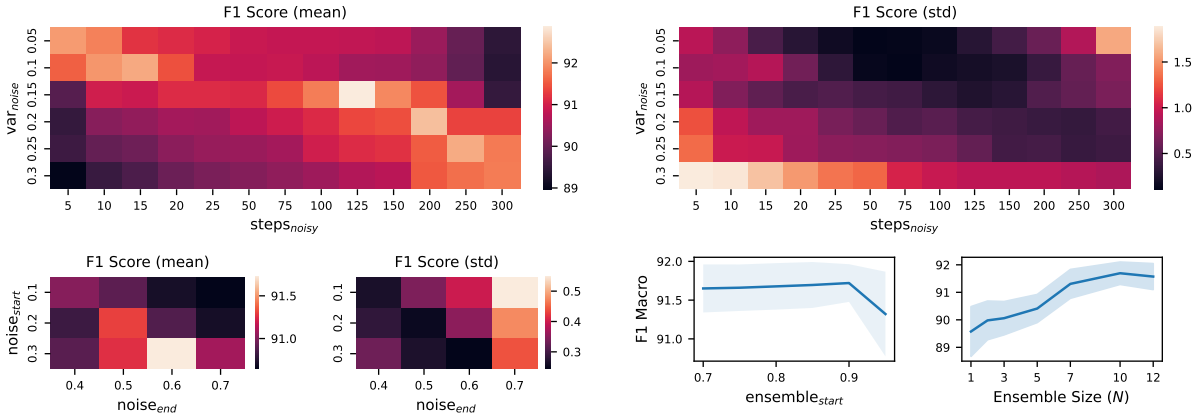

Figure 5: The effect of hyperparameter setup on the mitigation effectiveness of the DENI method, based on hyperparameter search for the BERT model on TREC dataset. The hyperparameters that affect each other are grouped together.

highest influence. Increasing the number of models that are used in the ensemble leads to significant increase in mitigation effectiveness (increasing performance and reducing deviation), but also a significant increase in the cost. However, after a certain number (10), the improvement of mitigation strength is negligible. As such, we recommend using $8 - 10$ models in the ensemble, as it represents the optimal trade-off between computation and mitigation.

When increasing the size of noise, we also need to add it less often (and vice versa) to get optimal performance, in terms of both performance and deviation. Small noise should be added often, while large noise should be added only occasionally. On the other hand, adding large noise often leads to large decrease in overall performance, while adding small noise only sporadically leads to no mitigation. We find that the optimal value for both factors is at their midpoint – adding noise of medium size (0.15) every so often (125 steps). However, this only holds when using the scaling factor. **When foregoing the scaling factor, we observe a significant sensitivity to the size of noise**.

Noise-adding noise should be started later in the training. At the start of optimisation, we observe a higher sensitivity of the model to the added noise, negatively affecting the mitigation effectiveness (i.e., leading to smaller performance increase and smaller decrease in deviation, often even being counterproductive). The sensitivity to the added noise slowly decreases as the training progresses (also due to the learning rate and scaling factor). As such, we observe that the optimal value for the $noise_{start}$ is after 30% of the training. At the same time, **adding noise for a longer period of time increases the overall deviation in results, while increasing the overall performance only until a certain point.** For example, starting the noise early and adding it for 60% of training leads to a large increase in deviation (and decrease in performance). Overall, optimal value for $noise_{end}$ is 30% of training steps after the $noise_{start}$ value, regardless of its value. **Finally, we observe negligible impact of the $ensemble_{start}$ hyperparameter.** The only requirement is to leave enough steps of optimisation for the model to deal with the added noise that can break the learned knowledge (i.e., leaving enough steps for the different models to converge to their optimal parameters). As such, it is enough to create the ensemble after 90% of training steps are already done, as creating the ensemble after this point leads to significant decrease mitigation (although if using larger number of steps, the start of ensemble may be delayed even further).

Overall, the hyperparameter sensitivity of the *DENI* method is not as significant. In majority of the cases, the optimal parameters can be determined based on heuristics (e.g., adding large noise often breaks the models), while slight deviation from the optimal values does not leave to a significant changes in the mitigation effectiveness. However, this holds true only when using the recommended hyperparameter setup introduced in Section 3 (using the scaling factor based on standard deviation of the individual parameters in the model and adding noise only to the newly initialised parameters).

C Experimental Setup: Further Details

C.1 General Experimental Setup

For the experiments, we are using English only datasets from the GLUE (Wang et al., 2018) benchmark suite and other publicly available datasets. The datasets we use from GLUE benchmark are: 1) SST2 (Socher et al., 2013), binary dataset with up to 68 000 labelled samples; 2) CoLA (Warstadt et al., 2019), binary dataset with up to 10 000 labelled samples; and 3) MRPC (Dolan and Brockett, 2005), binary dataset with up to 6 000 labelled samples. The remaining datasets are all multi-class datasets, specifically: 1) Ag News (Zhang et al., 2015) with 4 classes and up to 120 000 labelled samples; 2) TREC (Voorhees and Tice, 2000) with 6 classes and up to 6 000 labelled samples; 3) SNIPS (Coucke et al., 2018) with 7 classes and up to 15 000 labelled samples; and 4) DBPedia (Lehmann et al., 2015) with 14 classes and up to 630 000 labelled samples. All datasets are split into training and testing using 80-20 split. In addition, the training dataset is split into training and validation set, with the validation set used for hyperparameter tuning. When selecting the 1 000 labelled samples for training for the main experiments (or smaller/larger number of samples for the size change experiments), we use uniform split, i.e., we select the same number of labelled samples for each class.

Each experiment is repeated 20 times, each time with different random seeds that influences the randomness originating from model initialisation, data shuffling and non-deterministic operations in the model (e.g., dropout). Each such repeat uses the same set of training and testing data, mainly in order to make the data augmentation easier (i.e., we can use the augmentation only single time on the set of labelled samples without worrying about introducing an information leak, such as using for training samples that are paraphrases of the ones in the test or validation set).

For the experiments, we use the base versions of 3 representative pretrained language models (obtained from HuggingFace (Wolf et al., 2019)), specifically: 1) BERT (Devlin et al., 2019)[2] with 110M parameters; 2) RoBERTa (Liu et al., 2019)[3] with 130M parameters; and 3) ALBERT (Lan et al., 2020)[4] with 12M parameters. We add a Dropout layer with a drop rate of 0.3 followed by fully-connected classification layer on top of each of these models. Each model is trained for 10 epochs using Adam optimiser, learning rate of 1e-5 and batch size of 8 (with further modifications to these hyperparameters based on the mitigation strategy used, more information in Appendix C.2).

Besides full fine-tuning, where we train all parameters of the models, we also perform the experiments with parameter-efficient fine-tuning (PEFT) methods, specifically: 1) LoRA (Hu et al., 2022), applied to all attention layers, with rank 64, alpha 64, dropout 0.1, with bias for all layers and using rank stabilisation (Kalajdzievski, 2023), that trains approximately 2% of the parameters; 2) IA3 (Liu et al., 2022), applied to all atention and linear layers, that trains approximately 0.5% of parameters; and 3) UniPELT (Mao et al., 2022), which combines LoRA (same as above), Pfeiffer Adapter (Pfeiffer et al., 2020), with reduction factor of 16, and Prefix-Tuning (Li and Liang, 2021)), with prefix-length of 25, that trains approximately 10% of the parameters. As specified in the Limitations section, we do not focus on any prompt-based PEFT methods (P-tuning (Liu et al., 2023), Prefix-tuning (Li and Liang, 2021) or Prompt-tuning (Lester et al., 2021)), as these were not designed for optimising models with classification heads (and in our preliminary experiments, the prompt-based PEFT methods achieved significantly lower performance). Each of the PEFT methods has slightly different hyperparameters for training, as we have observed a significant sensitivity of the individual PEFT methods to the hyperparameter setup (with LoRA showing lowest sensitivity and UniPELT showing highest sensitivity). Specifically, we use following setup: 1) for LoRA we use learning rate of 1e-4 for BERT and ALBERT models and 1e-5 for RoBERTa model; 2) for IA3 we use learning rate of 75e-4; and 3) for UniPELT we use learning rate of 1e-4 for BERT, 25e-5 for RoBERTa and 6e-5 for ALBERT.

All of the hyperparameters, for all the combination of model and fine-tuning approach, are set using a separate hyperparameter optimisation using the validation data. This hyperparameter optimisation is done in a two-level fashion. First, the optimisation is run using large differences in the hyperparameter values, to find the approximate set of hyperparameters that should provide good performance on the given dataset. In the second step, we explore the hyperparameter space around these

[2] https://huggingface.co/bert-base-uncased
[3] https://huggingface.co/roberta-base
[4] https://huggingface.co/albert/albert-base-v2

approximate hyperparameters, to find the optimal set of parameters. In addition, the performance in the hyperparameter search is evaluated from 5 repeated runs (changing the model initialisation, data order and model non-determinism, but not the labelled training or validation data), taking their average into consideration. When choosing the hyperparameter values in the first level, we draw inspiration from related work, using the optimal parameters reported in papers that propose, or use these approaches (such as (Dodge et al., 2020; McCoy et al., 2020; Mosbach et al., 2021; Sellam et al., 2022). However, we also search through additional hyperparameter values besides those reported in related works to better explore the parameter space and obtain as precise results from the investigation as possible.

For the size change experiments (Section 4.3) the experimental setup is slightly different. Instead of choosing the 1 000 labelled samples, we instead choose N samples per class (called N-shots) and change the value of N, going from a low value up to the large part of the dataset. Specifically, we cover following N values: 1, 5, 10, 15, 20, 25, 30, 40, 50, 100, 150, 200, 300, 400, 500, 600, 700, 800, 900 and 1000. Due to the large computation cost of the size change experiment, the experiment is done only for the full fine-tuning of the BERT model on the TREC dataset. The remaining experimental setup is kept the same as for the main experiments.

Similarly, the experimental setup for the hyperparameter sensitivity analysis (Appendix B) is slightly different. Due to the large number of combinations, we limit the number of repeated runs only to 5. In addition, we run the sensitivity analysis only for the full fine-tuning of the BERT model on the TREC dataset. The hyperparameters and their values we search through are: 1) var_{noise} with values of 0.05, 0.1, 0.15, 0.2, 0.25 and 0.3; 2) $step_{noisy}$ with values of 5, 10, 15, 20, 25, 50, 75, 100, 125, 150, 200, 250 and 300; 3) $noise_{start}$ with values of 0.1, 0.2 and 0.3; 4) $noise_{end}$ with values of 0.4, 0.5, 0.6 and 0.7; 5) $ensemle_{start}$ with values of 0.7, 0.75, 0.8, 0.85, 0.9 and 0.95; and 6) ensemble size with values of 1, 2, 3, 5, 7, 9, 10 and 12. The remaining experimental setup is kept the same as for the main experiments.

C.2 Mitigation Strategies Setup

In this section, we provide further details regarding the hyperparameter setup for the individual mitigation strategies and how this affects their cost (represented as the number of steps). The hyperparameter setup for each of the methods, except for the baselines (*Default* and *Full*) and our method is determined using a hyperparameter search (with the same setup as for individual models, i.e., using average over 5 repeated runs on validation data).

The *Default* baseline uses the setup as specified in Appendix C.1, i.e., training for 10 epochs on 1 000 samples, with batch size of 8 and Adam optimiser. As such, the number of steps for each dataset is 1250, which we use as the normalisation number for the remaining mitigation strategies (i.e., having normalised cost value of 1).

The *All Data* baseline uses all the labelled samples available in the dataset, with the same setup as *Default* baseline. The exception are the AG News and DBPedia dataset, which are downsampled to use up to 55 000 samples. As such, the number of steps (and the cost of this baseline) is dataset dependent. For the TREC and MRPC datasets, the cost is 5 times the one of *Default*, for the CoLA the cost is 10, for SNIPS 11.5 and for the AG News, SST2 and DBPedia the cost is 55 times the *Default*.

The *Best Practices* mitigation strategy uses the AdamW optimiser with bias correction, warmup for 10% of training step followed by linear scheduler and increases the number of training epoch to 20. As such, the cost is 2 times the one in *Default* baseline.

The *Ensemble* mitigation strategy uses the same setup as the *Default* baselines, but trains 10 models with different initialisation and data order. As such, the cost is 10 times the one in *Default* baseline.

The $Noise_{Input}$ mitigation strategy introduces a Gaussian noise with the variance of 0.15 every 25 steps into the input embeddings of the model. Similarly, the $Noise_{Weights}$ mitigation strategy introduces a Gaussian noise with the variance of 0.15 (using scaling based on parameter standard deviation and number of steps) every 25 steps to all the randomly initialised parameters of the model (i.e., only the classification head or also the parameters added by PEFT method). As this does not modify the number of steps, the cost is the same as *Default* baseline.

For the *Stochastic Weight Averaging (SWA)* mitigation strategy we start the averaging after 25% of training steps using the same small learning rate as used for the model (as recommended by Lu et al. (2022)). As such, for the 75% of training, we perform optimisation on 2 models and so the cost is 1.75 time the one in *Default* baseline.

Dataset	Prompt
AG News, TREC, SNIPS, DBPedia, SST2	Rephrase an original question or statement 10 times. Format your responses like: 1. rephrase 1, 2. rephrase 2, ... , 10. rephrase 10. Original phrase: "{}".
MRPC	Paraphrase text consisting of two sentences in the format "Sentence 1: ... ; Sentence 2:..." Paraphrases both sentences 10 times. Format your responses in a JSON format. The text: "{}"
CoLA	Rephrase an original grammatically correct/incorrect text 10 times. Ensure that the rephrases are grammatically correct/incorrect. Format your responses like: 1. rephrase 1, 2. rephrase 2, ... , 10. rephrase 10. Original text: "{}".

Table 3: Prompts used for generating the paraphrases for the different datasets. The MRPC and CoLA dataset use a modified prompt to guarantee that the generated paraphrases are valid. For the CoLA dataset, the prompt guarantees that the paraphrase has the same class for the sample (either grammatically acceptable or not). For the MRPC dataset, the prompt guarantees that the two sentences are paraphrased at once, reducing the number of queries needed otherwise.

The *Mixout* mitigation strategy replaces the Dropout layers with mix probability of 0.9 as recommended by Lee et al. (2020). As it does not introduce any increase in the number of steps, the cost is the same as the one in *Default* baseline.

For the *Augment N* mitigation strategy, we follow the experimental setup from Cegin et al. (2023, 2024). A pre-trained large language model is used to paraphrase all the training samples 10 times and the generated paraphrases are then use as additional samples for training. For paraphrasing we use the ChatGPT 3.5 (version *gpt-3.5-turbo-0125*) model, with temperature value of 1 and other parameters set to default values. The prompts used to generate the paraphrases are in Table 3. For MRPC dataset, we formatted the augmented phrases in JSON for easier parsing. For CoLA dataset, we noticed that the resulting paraphrases for the grammatically incorrect sentences were in some cases (10% of the time) grammatically correct, meaning the label was incorrect. We did not do any additional manual filtering though (as it would be costly) and used the method as is with the all of the collected data. To reduce the required computation (and other) costs of the augmentation, we run it only a single time, asking the large language model to generate 10 paraphrases at once. When selecting what paraphrases to use for different N, we always select from top ones, i.e., when selecting 1 paraphrase, we always take the first generated by the model, or when selecting 3, we always take top three paraphrases. In addition, we increase the number of epochs to 12, as we have observed this significantly improves the performance. As the mitigation strategy introduces new samples and we increase the number of steps, the resulting cost of the strategy can be calculated as $1.2 * (N + 1)$.

Finally, for our method, we set the hyperparameters based on the hyperparameter search (Appendix B), superficially, we use var_{noise} set to 0.15, $steps_{noisy}$ set to 125, $noise_{start}$ set to 30% of training steps; $noise_{end}$ set to 60% of the training steps; $ensemble_{start}$ set to 90% of the training steps; ensemble size of 10; and the remaining parameters as recommended in Section 3 (i.e., using Gaussian noise introduced only to newly initialised parameters, same number of steps for $steps_{noisy}$ and $steps_{regular}$, using scaled noise based on the parameters and also on number of already done steps for the Noisy Interpolation part). In addition, for the *DENIALS* configuration, we use the *Augment 1* to generate the new samples. We calculate the cost for the *DENI* method in following way. For the Delayed Ensemble (DE), a single model is trained for 90% of the training steps (cost of 0.9) and 10 models are trained for the remaining training steps (cost of $10 * 0.1$) resulting in a cost of 1.9. For the Noisy Interpolation part a single model is trained for 70% of the training steps (cost of 0.7), and the noise is added for 30% of training steps, from which we train the ensemble for 10% of training (cost of 1), single model for additional 10% of training (cost of 0.1) and another ensemble for additional 10% of training (cost of 1), resulting in cost of 2.8. The cost of the Delayed Ensemble with Noisy Interpolation (DENI) is the combination of the Noisy Interpolation (2.8) and the cost for the final ensemble (0.9) resulting in cost of 3.7. Finally for the Delayed Ensemble with Noisy Interpolation and Augmented Labelled Samples (DENIALS) we double the number of steps and so the cost is calculated as $(N + 1) * 3.7$, resulting in cost of 7.4.

D Additional Results: Relationship Between Performance, Deviation and Normalised Cost

In this Appendix, we provide a visualisation of the relationship between the performance, standard deviation and the normalised cost for the different mitigation strategies. The visualisation serves to allow for better evaluation and comparison for the benefit of the mitigation strategies, which also takes cost into consideration. The relationship is visualised in Figure 6 for the BERT model for all fine-tuning methods across all datasets.

We observe that the mitigation strategies that provide the highest benefit, increasing performance and reducing standard deviation, also introduce increase in the computation cost. When taking the cost into consideration, the *DENI* method (without the augmented labelled samples) appears the most efficient, as it provides similar (and often higher) performance, similar (and often lower) standard deviation, while requiring only half of the computation costs. Similarly, the *DENI* method provides higher performance and similar (and often lower) standard deviation than the *Ensemble* mitigation strategy, while requiring only a 1/3 of the computation cost. An additional interesting comparison is with the *All Data* baseline, where the *DENI* method, but often also *Ensemble* mitigation strategy, appear close to the baseline, often outperforming it in terms standard deviation.

E Additional Results: *Augment N* With Higher Number of Paraphrases

In this Appendix, we provide the results for the *Augment N* mitigation strategy with additional values for N, specifically 3, 5 and 10. The results are reported for all the models and datasets and are normalised by the mean value of *Default* baseline for each of the given datasets. For the BERT model, the results are reported in Figure 7 for full fine-tuning, Figure 8 for LoRA, Figure 9 for IA3 and Figure 10 for UniPELT. For the RoBERTa model, the results are reported in Figure 11 for full fine-tuning, Figure 12 for LoRA, Figure 13 for IA3 and Figure 14 for UniPELT. For the ALBERT model, the results are reported in Figure 15 for full fine-tuning, Figure 16 for LoRA, Figure 17 for IA3 and Figure 18 for UniPELT.

For the majority of the cases, the highest performance increase is observed when using only 1 or 2 paraphrases for each of the labelled samples (the performance reported in the main content of the paper). Increasing the number of paraphrases, the performance gradually decreases, achieving lowest performance with 10 paraphrased samples. The reason for this decrease can be twofold. First, we run the paraphrasing only a single time, generating 10 paraphrases at once and selecting from the top. As the model tries to generate 10 paraphrases at once, the later ones may be significantly different, using more obscure words. As such, adding such paraphrases into the training may confuse the model as the distribution of words is significantly different (and may even represent noise for the model). Second, with the higher number of paraphrases, the model may start to lose its generalisability as it is trained on very similar samples (i.e., paraphrases) and thus start overfitting the data distribution which is not representative of the test data. However, we observe some exceptions, where using full set of paraphrases does not lower the performance.

In addition, we observe that the benefit of data augmentation is dependent on the model and the fine-tuning approach used. When limiting the number of trainable parameters, either by using the IA3 PEFT method (which trains approximately 0.5% of the parameters) or using the ALBERT model (which has only 12M parameters), we observe a significantly higher benefit of the data augmentation. In specific cases, using the augmented data can lead to a performance higher than from any other mitigation strategy, such as using ALBERT model trained with IA3 method on CoLA dataset (see Table 6).

Finally, we can observe that in specific cases, the augmented data cause significant problems to the models. Specifically, we observe a significant decrease in performance for the UniPELT method on the MRPC dataset across all models. In addition, we observe that the augmented data on MRPC dataset show higher standard deviation across all models and fine-tuning approaches. The reason for both may be due to the characteristics of the MRPC dataset – the task of determining the semantic similarity between 2 sentences. Although we specifically design the prompt to paraphrase both sentences at once, which should keep their semantic similarity, the large language model may not produce correct paraphrases in this case (e.g., introducing a significant changes, making the sentences too dissimilar). Such augmented data may cause problems to the models and especially to UniPELT, which we observed to be the most brittle

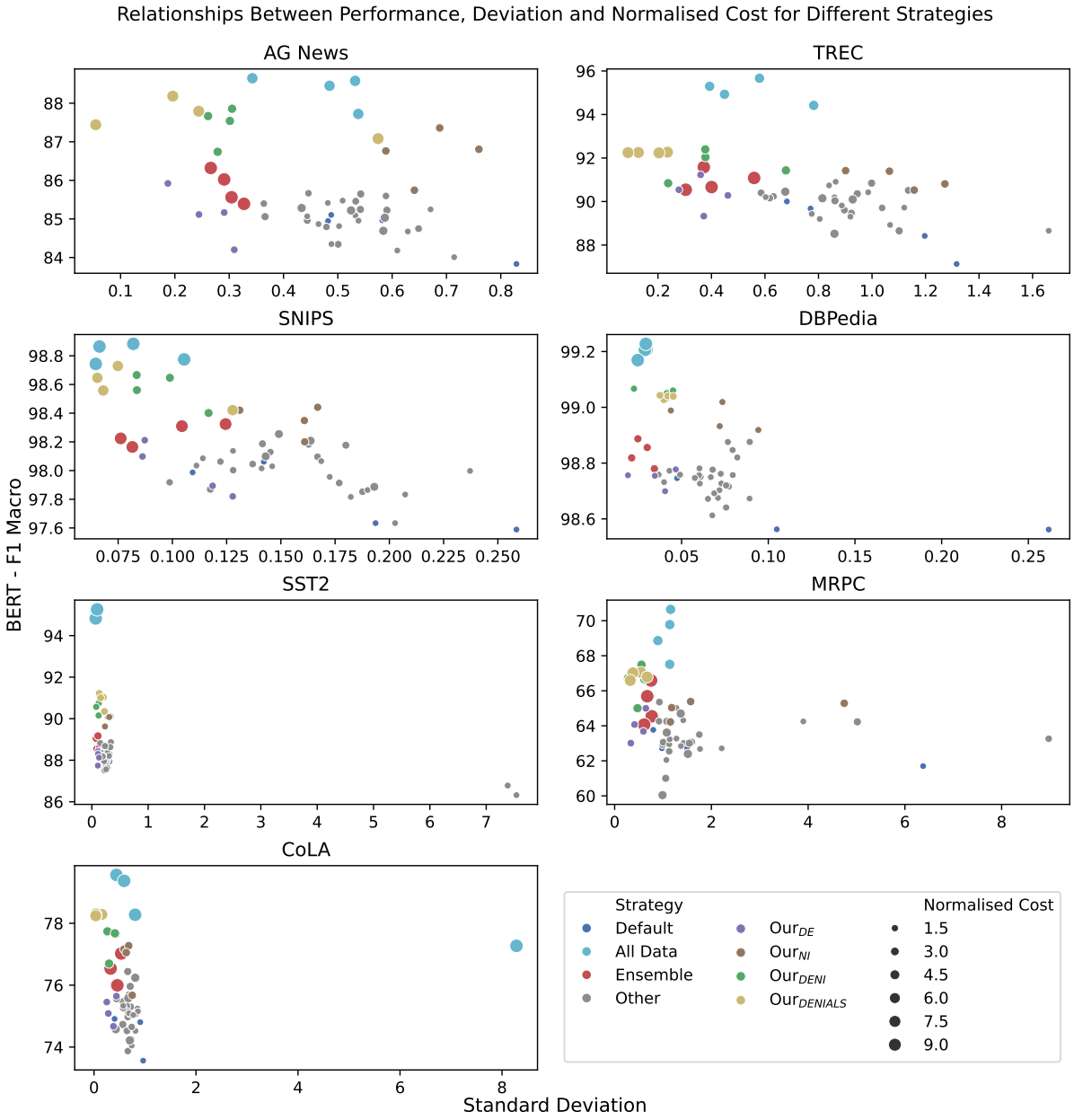

Figure 6: The relationship between performance, standard deviation and the normalised cost for the different mitigation strategies. The results are presented for the BERT model and all fine-tuning methods on all datasets. For the sake of better readability, we highlight only the most efficient mitigation strategies and baselines (highest performance and lowest standard deviation), greying out all the remaining ones. We can observe that the *DENI* method provides the most effective mitigation, with the performance and standard deviation being on par or better than other strategies, while requiring only a fraction of their computation costs.

fine-tuning approach.

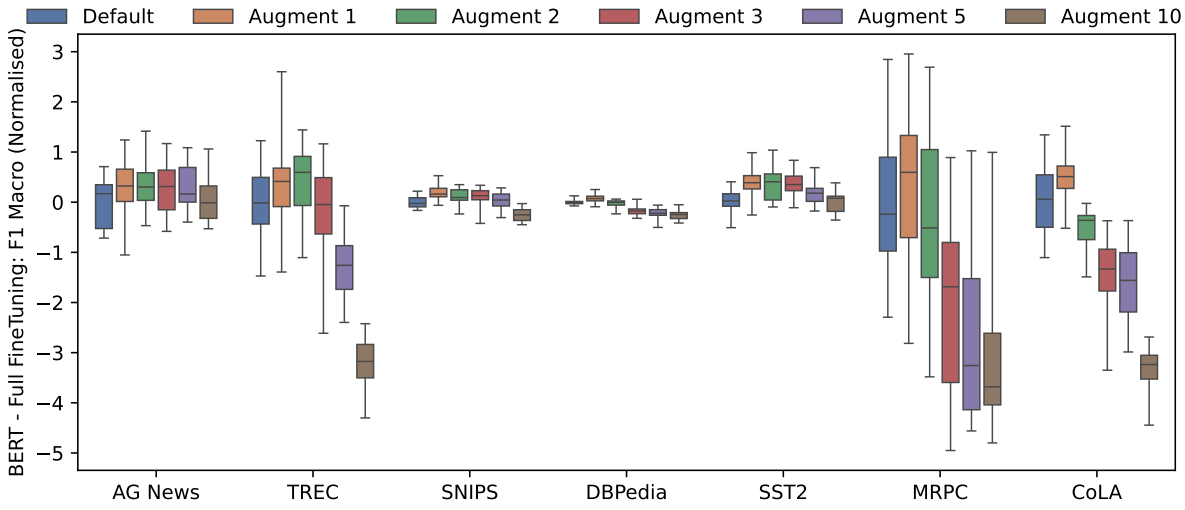

Figure 7: The performance of the Augment mitigation strategy for the different values of N for BERT using full fine-tuning across all datasets. The reported performance is normalised using the mean value of the *Default* baselines for each of the given datasets (i.e., subtracting the mean from all values).

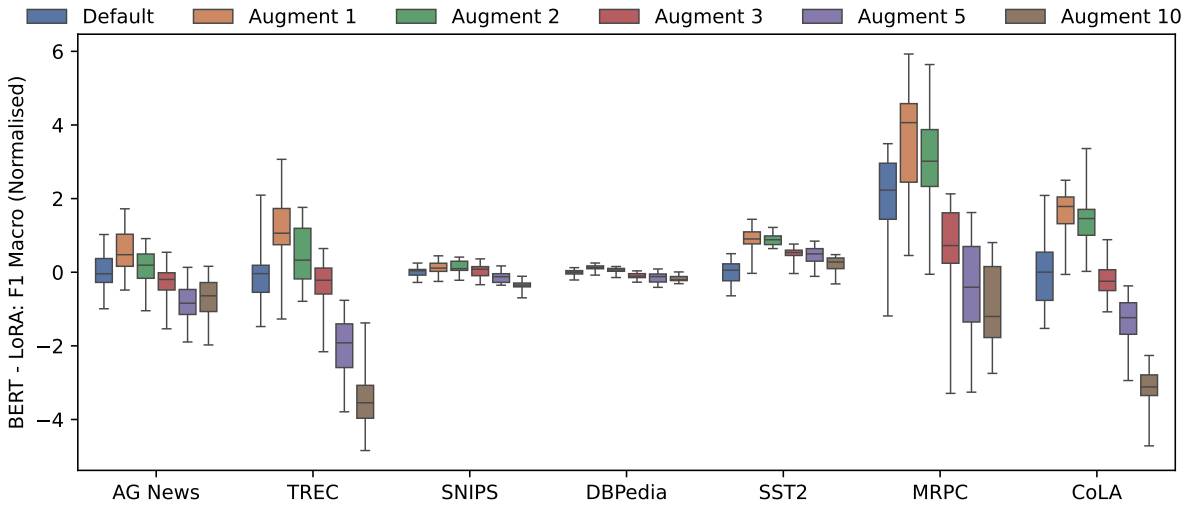

Figure 8: The performance of the Augment mitigation strategy for the different values of N for BERT using LoRA across all datasets. The reported performance is normalised using the mean value of the *Default* baselines for each of the given datasets (i.e., subtracting the mean from all values).

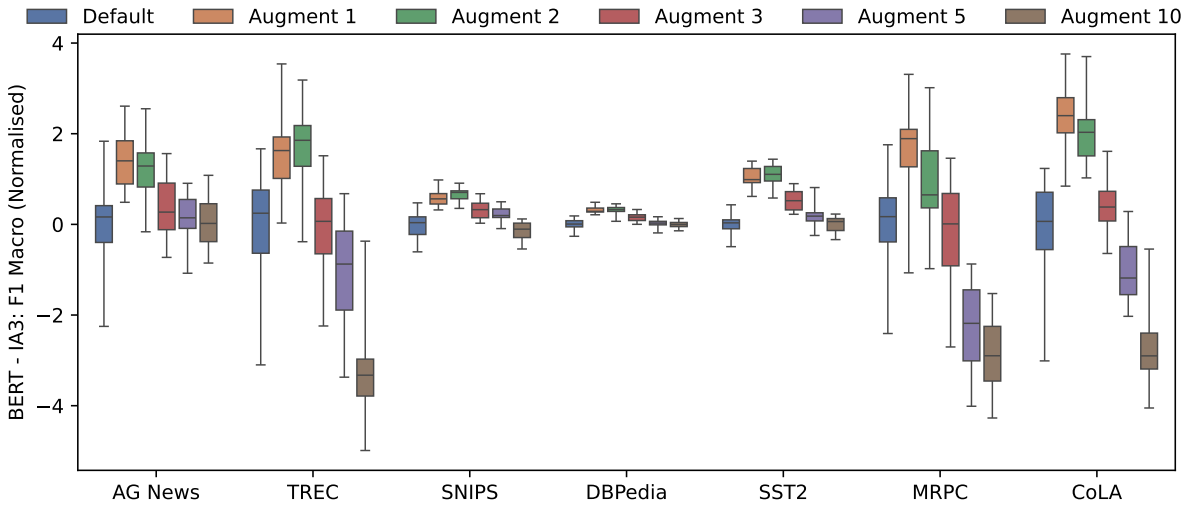

Figure 9: The performance of the Augment mitigation strategy for the different values of N for BERT using IA3 across all datasets. The reported performance is normalised using the mean value of the *Default* baselines for each of the given datasets (i.e., subtracting the mean from all values).

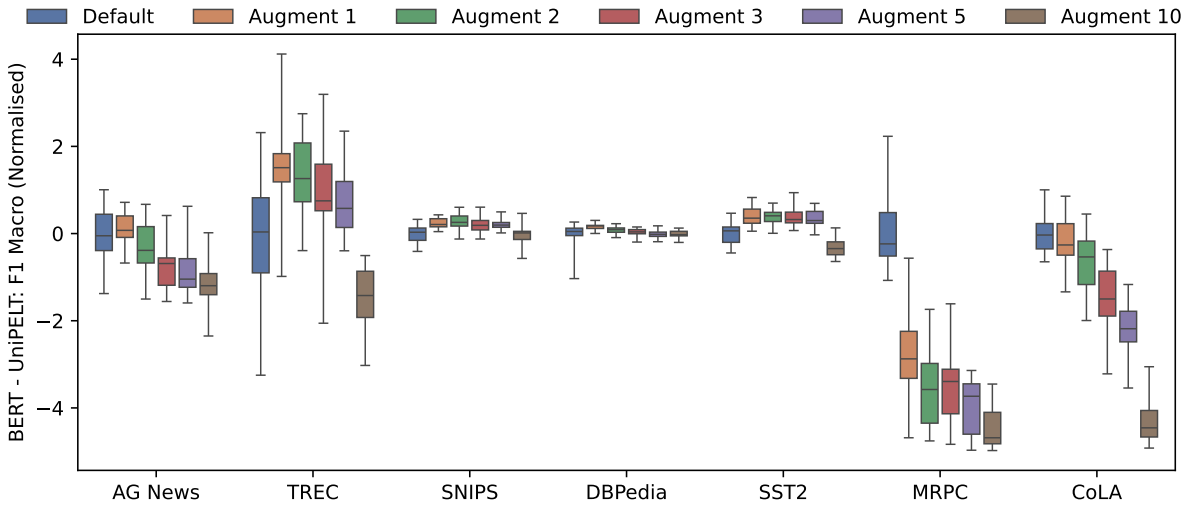

Figure 10: The performance of the Augment mitigation strategy for the different values of N for BERT using UniPELT across all datasets. The reported performance is normalised using the mean value of the *Default* baselines for each of the given datasets (i.e., subtracting the mean from all values).

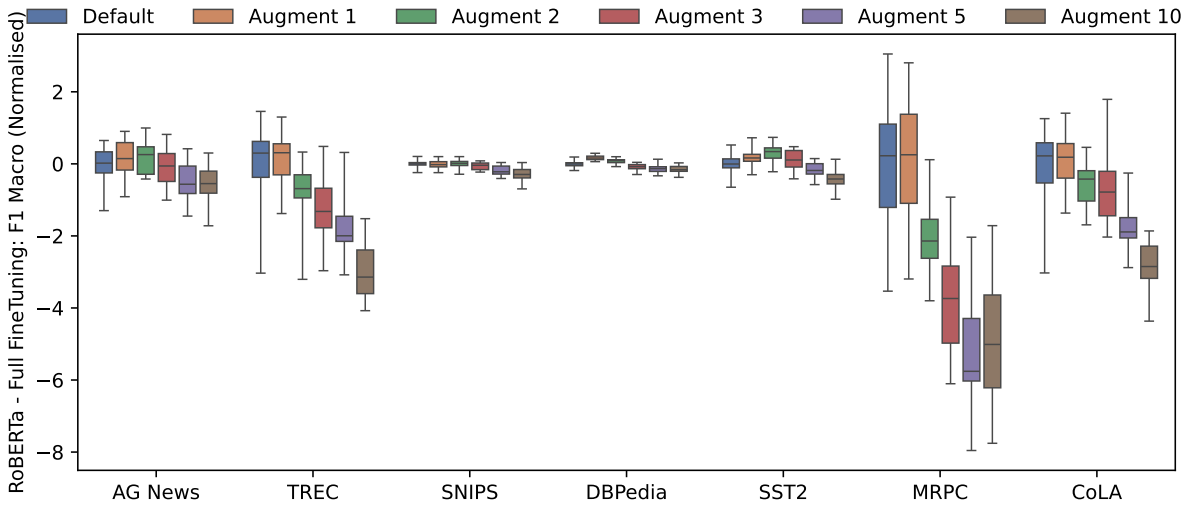

Figure 11: The performance of the Augment mitigation strategy for the different values of N for RoBERTa using full fine-tuning across all datasets. The reported performance is normalised using the mean value of the *Default* baselines for each of the given datasets (i.e., subtracting the mean from all values).

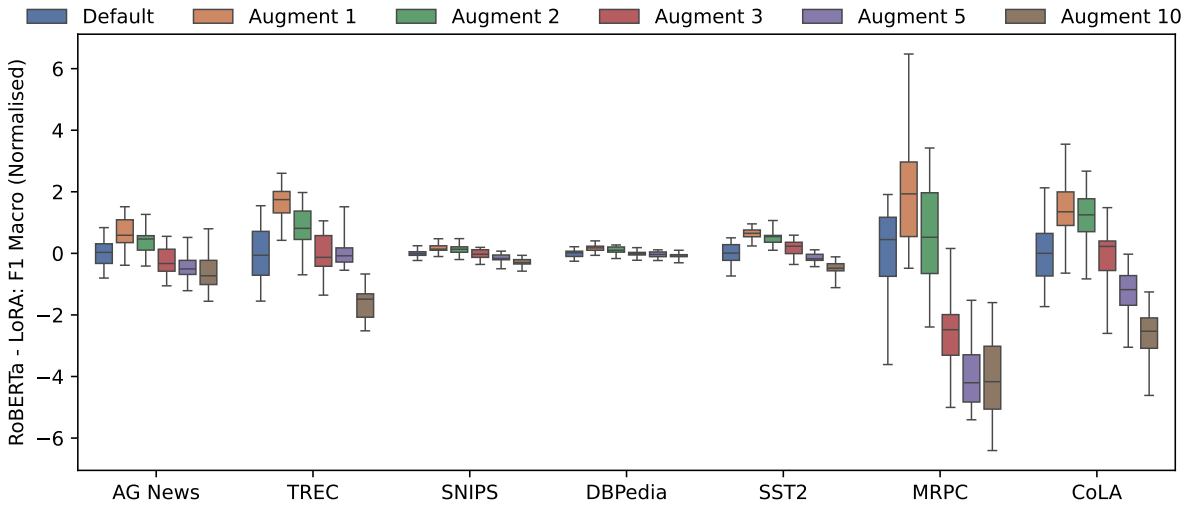

Figure 12: The performance of the Augment mitigation strategy for the different values of N for RoBERTa using LoRA across all datasets. The reported performance is normalised using the mean value of the *Default* baselines for each of the given datasets (i.e., subtracting the mean from all values).

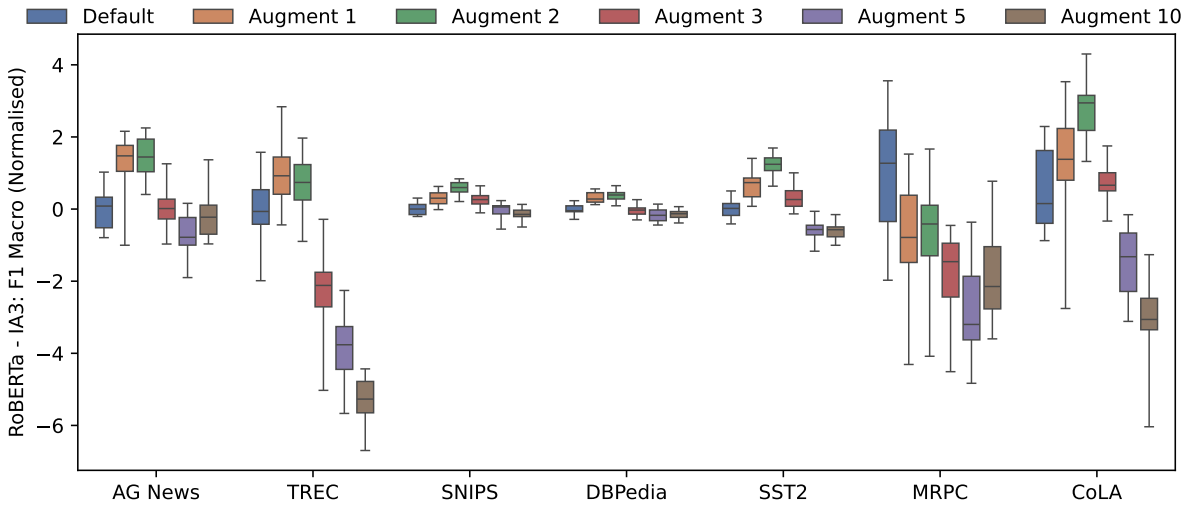

Figure 13: The performance of the Augment mitigation strategy for the different values of N for RoBERTa using IA3 across all datasets. The reported performance is normalised using the mean value of the *Default* baselines for each of the given datasets (i.e., subtracting the mean from all values).

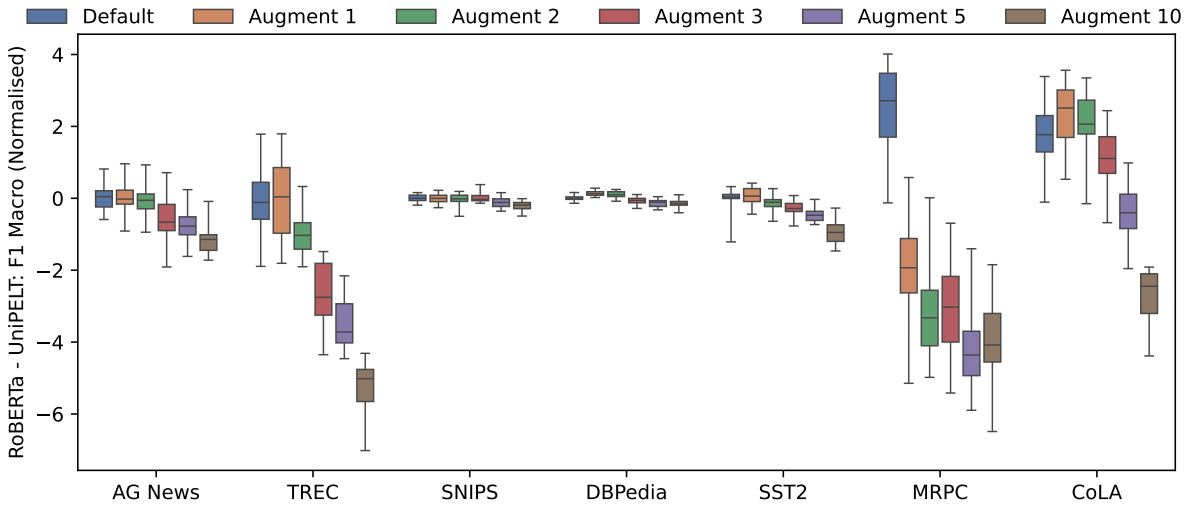

Figure 14: The performance of the Augment mitigation strategy for the different values of N for RoBERTa using UniPELT across all datasets. The reported performance is normalised using the mean value of the *Default* baselines for each of the given datasets (i.e., subtracting the mean from all values).

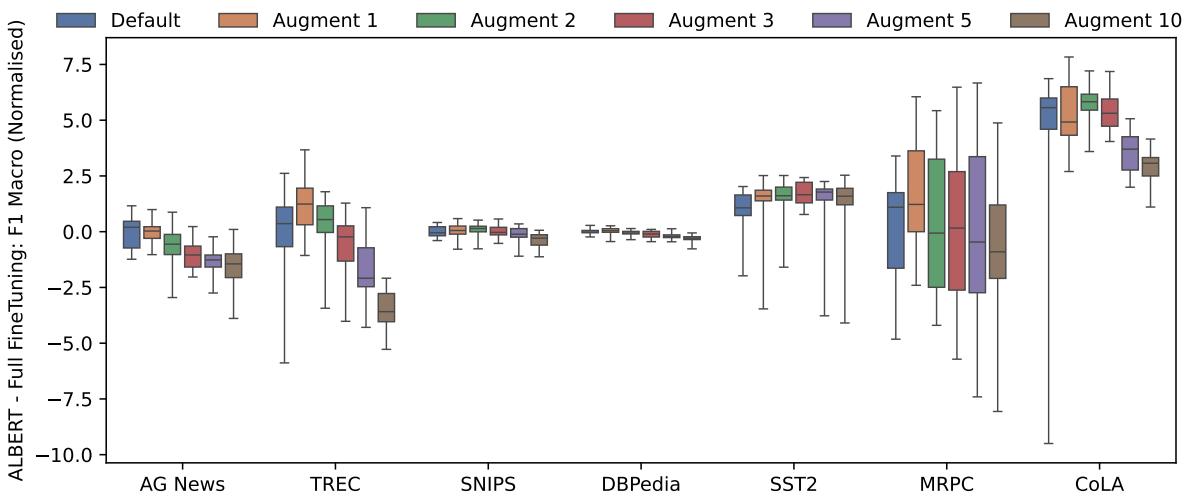

Figure 15: The performance of the Augment mitigation strategy for the different values of N for ALBERT using full fine-tuning across all datasets. The reported performance is normalised using the mean value of the *Default* baselines for each of the given datasets (i.e., subtracting the mean from all values).

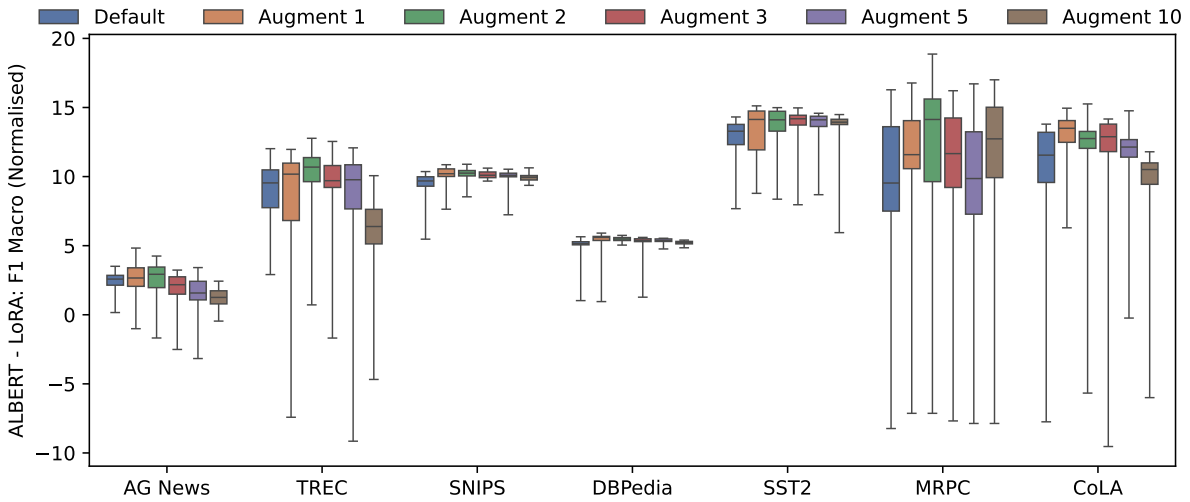

Figure 16: The performance of the Augment mitigation strategy for the different values of N for ALBERT using LoRA across all datasets. The reported performance is normalised using the mean value of the *Default* baselines for each of the given datasets (i.e., subtracting the mean from all values).

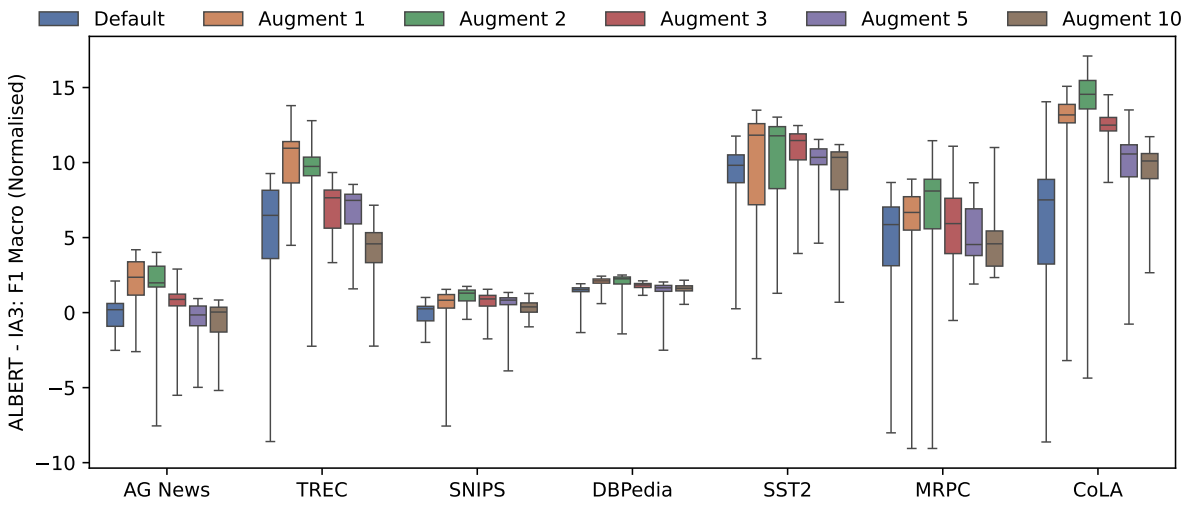

Figure 17: The performance of the Augment mitigation strategy for the different values of N for ALBERT using IA3 across all datasets. The reported performance is normalised using the mean value of the *Default* baselines for each of the given datasets (i.e., subtracting the mean from all values).

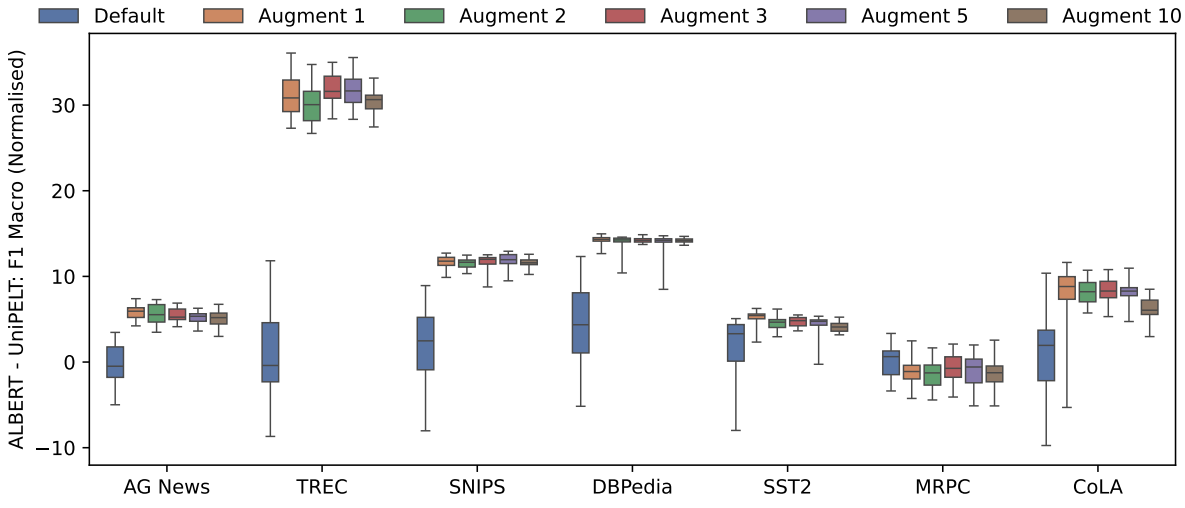

Figure 18: The performance of the Augment mitigation strategy for the different values of N for ALBERT using UniPELT across all datasets. The reported performance is normalised using the mean value of the *Default* baselines for each of the given datasets (i.e., subtracting the mean from all values).

F Additional Results: All Datasets, Models and Fine-Tuning Approaches

In this Appendix we provide the results for each combination of dataset, model, mitigation strategy and fine-tuning approach. All the results are provided in Table 4 for the BERT model, in Table 5 for the RoBERTa model and in Table 6 for the ALBERT model. In addition, we provide results in a form of figures (boxplots) separately for each dataset and model in a following way:

- For BERT model in Figure 19 for AG News dataset, Figure 20 for TREC dataset, Figure 21 for SNIPS dataset, Figure 22 for DBPedia dataset, Figure 23 for SST2 dataset, Figure 24 for MRPC dataset and Figure 25 for CoLA dataset.
- For RoBERTa model in Figure 26 for AG News dataset, Figure 27 for TREC dataset, Figure 28 for SNIPS dataset, Figure 29 for DBPedia dataset, Figure 30 for SST2 dataset, Figure 31 for MRPC dataset and Figure 32 for CoLA dataset.
- For ALBERT model in Figure 33 for AG News dataset, Figure 34 for TREC dataset, Figure 35 for SNIPS dataset, Figure 36 for DBPedia dataset, Figure 37 for SST2 dataset, Figure 38 for MRPC dataset and Figure 39 for CoLA dataset.

For the figures, we omit the failed runs in the visualisations (i.e., runs that show significantly lower performance for the given strategy or than the average of all the other strategies, for example achieving F1 score of 1% instead of 80% on TREC dataset). As such, specific mitigation strategies may be missing in the figures in some cases – in such case, all the runs using the mitigation strategies resulted in failed runs, such as Mixout on MRPC or CoLA dataset with RoBERTa model trained using IA3, or the *Default* baseline on TREC dataset with ALBERT model trained using UniPELT.

When it comes to the further models and dataset, we observe only minor differences in the results compared to the ones reported in the main body of the paper. We summarise the main findings and potential differences below.

On the different datasets, we observe different benefit for the individual mitigation strategies. For example, for the BERT model trained using UniPELT, the *SWA* mitigation strategy achieves similar performance to the *Ensemble* mitigation strategies on the AG News, MRPC or CoLA datasets, while on SNIPS, DBPedia or TREC it leads to lower benefit. Similar behaviour can be observed for other methods as well. The data augmentation provides significantly larger benefit when the number of trainable parameters is small, such as when using IA3 or the ALBERT model. For example, the *Augment N* strategy outperforms the *Ensemble* strategy in these cases. In addition, this has also the effect on our proposed method. The *DENI* method (or *DENIALS* method) achieves the highest performance increase and standard deviation reduction across all datasets, models and fine-tuning approaches, with the exception of RoBERTa and ALBERT trained using IA3 on CoLA dataset, where the *Augment 2* strategy provides the most benefit. This can be explained by: 1) the benefit of augmented samples when the number of trainable parameters is small; and 2) we observe that the augmented samples for CoLA dataset appear to provide larger benefit across all models and fine-tuning approaches. As such, in these cases the benefit of adding further samples through augmentation outweighs benefit of regular mitigation strategies.

In addition, we observe different level of standard deviation for the different models, especially for the ALBERT model. As such, we also observe a different benefit of the individual mitigation strategies on these models as compared to BERT. For example, we observe higher benefit of the *Stochastic Weight Averaging*, even outperforming the *Ensemble* mitigation strategy and all of our proposed methods when using ALBERT model trained using IA3 on the CoLA dataset. At the same time, the *Ensemble* strategy and the *DENI* method provide more consistent and more significant reduction in the standard deviation on the ALBERT and RoBERTa models across the different datasets and fine-tuning approaches, while the majority of the remaining mitigation strategies often show significantly lower consistency, often not leading to any mitigation benefit over the *Default* baseline. As such, we can better observe the efficiency and effectiveness of the different mitigation strategies on these cases – with *Ensemble* and our *DENI* method being significantly more effective and efficient.

The different components of the *DENI* method provide the same benefit across all the models, datasets and fine-tuning methods. The *Delayed Ensemble (DE)* reduces the standard deviation, achiev-

BERT	AG NEWS	TREC	SNIPS	DBPEDIA	SST2	MRPC	COLA
			FULL FINETUNING				
DEFAULT	$84.95_{0.482}$	$90.00_{0.682}$	$97.99_{0.109}$	$98.75_{0.047}$	$88.27_{0.230}$	$62.73_{1.497}$	$75.11_{0.662}$
ALL DATA	$88.65_{0.343}$	$95.66_{0.579}$	$98.86_{0.066}$	$99.21_{0.030}$	$95.15_{0.078}$	$68.86_{0.893}$	$79.56_{0.440}$
BEST PRACTICES	$84.96_{0.444}$	$90.40_{0.585}$	$98.05_{0.137}$	$98.76_{0.049}$	$88.32_{0.313}$	$63.07_{1.593}$	$75.30_{0.718}$
ENSEMBLE	$85.56_{0.304}$	$90.67_{0.400}$	$98.22_{0.076}$	$98.82_{0.021}$	$89.05_{0.079}$	$65.69_{0.674}$	$76.54_{0.325}$
NOISE$_{Input}$	$85.07_{0.444}$	$89.43_{0.775}$	$98.09_{0.114}$	$98.77_{0.043}$	$88.25_{0.285}$	$62.44_{1.499}$	$75.05_{0.817}$
NOISE$_{Weights}$	$85.25_{0.671}$	$90.20_{0.860}$	$98.03_{0.146}$	$98.75_{0.061}$	$88.04_{0.266}$	$62.68_{1.765}$	$75.06_{0.699}$
SWA	$85.40_{0.364}$	$90.24_{0.632}$	$98.06_{0.122}$	$98.76_{0.060}$	$88.54_{0.211}$	$63.50_{1.751}$	$75.26_{0.572}$
MIXOUT	$84.96_{0.538}$	$90.20_{0.603}$	$98.03_{0.111}$	$98.73_{0.040}$	$88.39_{0.280}$	$63.23_{1.137}$	$75.33_{0.572}$
AUGMENT 1	$85.25_{0.542}$	$90.36_{0.945}$	$98.19_{0.142}$	$98.82_{0.082}$	$88.66_{0.286}$	$63.01_{1.544}$	$75.56_{0.443}$
AUGMENT 2	$85.28_{0.433}$	$90.45_{0.676}$	$98.10_{0.143}$	$98.72_{0.076}$	$88.63_{0.327}$	$62.40_{1.514}$	$74.58_{0.426}$
OUR$_{DE}$	$85.17_{0.291}$	$90.54_{0.278}$	$98.10_{0.086}$	$98.76_{0.019}$	$88.43_{0.107}$	$63.68_{0.594}$	$75.09_{0.281}$
OUR$_{NI}$	$86.81_{0.760}$	$91.42_{0.901}$	$98.42_{0.131}$	$98.99_{0.044}$	$90.08_{0.284}$	$65.38_{1.571}$	$77.16_{0.586}$
OUR$_{DENI}$	$87.67_{0.261}$	$92.04_{0.377}$	$98.65_{0.099}$	$\mathbf{99.07}_{0.023}$	$90.82_{0.129}$	$66.66_{0.613}$	$77.74_{0.263}$
OUR$_{DENIALS}$	$\mathbf{88.18}_{0.196}$	$\mathbf{92.27}_{0.236}$	$\mathbf{98.73}_{0.075}$	$99.03_{0.040}$	$\mathbf{91.22}_{0.132}$	$\mathbf{67.04}_{0.544}$	$\mathbf{78.29}_{0.153}$
			LORA				
DEFAULT	$85.10_{0.488}$	$89.68_{0.770}$	$98.06_{0.142}$	$98.72_{0.077}$	$87.95_{0.315}$	$61.70_{6.377}$	$74.80_{0.907}$
ALL DATA	$88.45_{0.485}$	$95.29_{0.393}$	$98.88_{0.082}$	$99.21_{0.029}$	$95.16_{0.086}$	$70.64_{1.156}$	$77.27_{8.283}$
BEST PRACTICES	$85.46_{0.533}$	$90.51_{1.135}$	$98.10_{0.167}$	$98.74_{0.058}$	$88.13_{0.263}$	$65.36_{0.925}$	$74.97_{0.662}$
ENSEMBLE	$86.32_{0.266}$	$91.58_{0.371}$	$98.32_{0.125}$	$98.86_{0.030}$	$89.17_{0.109}$	$66.59_{0.753}$	$77.02_{0.533}$
NOISE$_{Input}$	$85.42_{0.482}$	$90.42_{0.986}$	$98.07_{0.169}$	$98.76_{0.037}$	$88.16_{0.285}$	$65.02_{1.277}$	$75.48_{0.545}$
NOISE$_{Weights}$	$85.09_{0.532}$	$90.74_{0.840}$	$98.02_{0.141}$	$98.73_{0.061}$	$88.14_{0.280}$	$64.25_{3.900}$	$75.25_{0.851}$
SWA	$85.67_{0.446}$	$90.33_{0.948}$	$98.13_{0.145}$	$98.75_{0.058}$	$86.78_{7.381}$	$63.26_{8.973}$	$74.65_{0.739}$
MIXOUT	$85.47_{0.509}$	$90.91_{0.864}$	$98.14_{0.128}$	$98.68_{0.071}$	$88.21_{0.306}$	$64.31_{1.418}$	$75.15_{0.860}$
AUGMENT 1	$85.65_{0.542}$	$90.84_{0.998}$	$98.18_{0.163}$	$98.85_{0.079}$	$88.87_{0.339}$	$64.23_{5.017}$	$76.44_{0.662}$
AUGMENT 2	$85.22_{0.524}$	$90.15_{0.814}$	$98.21_{0.164}$	$98.78_{0.068}$	$88.82_{0.149}$	$64.70_{1.366}$	$76.24_{0.808}$
OUR$_{DE}$	$85.92_{0.187}$	$91.23_{0.359}$	$98.21_{0.087}$	$98.75_{0.035}$	$88.30_{0.111}$	$65.00_{0.644}$	$75.65_{0.439}$
OUR$_{NI}$	$87.36_{0.688}$	$91.39_{1.065}$	$98.44_{0.167}$	$98.93_{0.072}$	$90.10_{0.332}$	$65.29_{4.747}$	$77.28_{0.680}$
OUR$_{DENI}$	$\mathbf{87.86}_{0.305}$	$\mathbf{92.39}_{0.377}$	$\mathbf{98.67}_{0.084}$	$\mathbf{99.05}_{0.041}$	$90.74_{0.122}$	$\mathbf{67.49}_{0.554}$	$77.66_{0.425}$
OUR$_{DENIALS}$	$87.79_{0.244}$	$92.24_{0.204}$	$98.65_{0.065}$	$99.04_{0.038}$	$\mathbf{91.03}_{0.206}$	$66.79_{0.671}$	$\mathbf{78.24}_{0.041}$
			IA3				
DEFAULT	$83.83_{0.829}$	$88.42_{1.197}$	$97.59_{0.259}$	$98.56_{0.105}$	$87.60_{0.216}$	$62.72_{0.979}$	$73.56_{0.963}$
ALL DATA	$87.72_{0.538}$	$94.93_{0.449}$	$98.74_{0.065}$	$99.17_{0.025}$	$94.83_{0.069}$	$67.51_{1.139}$	$78.28_{0.806}$
BEST PRACTICES	$84.34_{0.501}$	$89.47_{0.923}$	$97.85_{0.188}$	$98.67_{0.065}$	$87.58_{0.225}$	$62.92_{0.994}$	$74.53_{0.646}$
ENSEMBLE	$85.39_{0.327}$	$90.54_{0.304}$	$98.16_{0.082}$	$98.78_{0.034}$	$88.55_{0.092}$	$64.54_{0.767}$	$75.99_{0.440}$
NOISE$_{Input}$	$84.18_{0.609}$	$89.30_{0.919}$	$97.83_{0.207}$	$98.61_{0.068}$	$87.51_{0.223}$	$62.05_{1.070}$	$74.52_{0.812}$
NOISE$_{Weights}$	$84.35_{0.488}$	$89.20_{0.805}$	$97.82_{0.182}$	$98.67_{0.089}$	$87.63_{0.277}$	$63.02_{1.426}$	$74.26_{0.739}$
SWA	$84.79_{0.479}$	$90.15_{0.618}$	$97.92_{0.099}$	$98.78_{0.060}$	$88.19_{0.196}$	$62.54_{1.127}$	$73.87_{0.663}$
MIXOUT	$84.01_{0.714}$	$88.66_{1.660}$	$97.63_{0.203}$	$98.70_{0.072}$	$87.56_{0.253}$	$62.71_{2.212}$	$74.05_{0.738}$
AUGMENT 1	$85.23_{0.590}$	$90.03_{0.861}$	$98.18_{0.180}$	$98.88_{0.077}$	$88.65_{0.229}$	$64.25_{1.079}$	$75.96_{0.710}$
AUGMENT 2	$85.03_{0.587}$	$90.10_{0.928}$	$98.25_{0.149}$	$98.88_{0.089}$	$88.68_{0.234}$	$63.61_{1.078}$	$75.58_{0.672}$
OUR$_{DE}$	$84.20_{0.309}$	$89.33_{0.371}$	$97.82_{0.128}$	$98.70_{0.040}$	$87.74_{0.107}$	$63.01_{0.335}$	$74.67_{0.381}$
OUR$_{NI}$	$85.74_{0.641}$	$90.53_{1.158}$	$98.20_{0.161}$	$98.92_{0.094}$	$89.62_{0.233}$	$64.22_{1.152}$	$75.67_{0.749}$
OUR$_{DENI}$	$86.74_{0.279}$	$90.84_{0.238}$	$98.40_{0.117}$	$99.03_{0.041}$	$90.15_{0.121}$	$65.00_{0.473}$	$76.70_{0.295}$
OUR$_{DENIALS}$	$\mathbf{87.08}_{0.574}$	$\mathbf{92.26}_{0.127}$	$\mathbf{98.42}_{0.128}$	$\mathbf{99.04}_{0.042}$	$\mathbf{90.35}_{0.224}$	$\mathbf{67.02}_{0.378}$	$\mathbf{78.30}_{0.037}$
			UNIPELT				
DEFAULT	$84.97_{0.582}$	$87.13_{1.316}$	$97.63_{0.194}$	$98.56_{0.262}$	$87.82_{0.233}$	$63.77_{0.798}$	$74.91_{0.403}$
ALL DATA	$88.58_{0.532}$	$94.42_{0.782}$	$98.77_{0.105}$	$99.23_{0.029}$	$95.26_{0.095}$	$69.77_{1.139}$	$79.37_{0.590}$
BEST PRACTICES	$84.75_{0.649}$	$89.61_{0.897}$	$97.91_{0.177}$	$98.76_{0.073}$	$88.01_{0.285}$	$63.06_{1.003}$	$75.32_{0.642}$
ENSEMBLE	$86.02_{0.291}$	$91.08_{0.559}$	$98.31_{0.104}$	$98.89_{0.025}$	$88.63_{0.141}$	$64.07_{0.610}$	$76.00_{0.458}$
NOISE$_{Input}$	$84.67_{0.629}$	$88.93_{1.068}$	$97.86_{0.190}$	$98.75_{0.067}$	$87.82_{0.247}$	$62.94_{1.128}$	$75.16_{0.652}$
NOISE$_{Weights}$	$84.87_{0.464}$	$89.72_{1.120}$	$98.00_{0.237}$	$98.73_{0.073}$	$86.32_{7.537}$	$63.27_{1.278}$	$75.09_{0.687}$
SWA	$85.60_{0.589}$	$89.71_{1.038}$	$98.00_{0.128}$	$98.76_{0.080}$	$88.37_{0.163}$	$64.26_{0.919}$	$75.69_{0.685}$
MIXOUT	$84.81_{0.502}$	$89.82_{0.887}$	$97.96_{0.173}$	$98.69_{0.069}$	$87.95_{0.217}$	$62.84_{1.371}$	$75.04_{0.770}$
AUGMENT 1	$85.06_{0.366}$	$88.65_{1.101}$	$97.87_{0.118}$	$98.72_{0.077}$	$88.21_{0.200}$	$61.00_{1.055}$	$74.73_{0.564}$
AUGMENT 2	$84.69_{0.584}$	$88.52_{0.860}$	$97.89_{0.193}$	$98.64_{0.076}$	$88.18_{0.192}$	$60.04_{0.990}$	$74.22_{0.702}$
OUR$_{DE}$	$85.11_{0.244}$	$90.28_{0.461}$	$97.89_{0.119}$	$98.78_{0.047}$	$88.13_{0.128}$	$64.07_{0.412}$	$75.46_{0.250}$
OUR$_{NI}$	$86.76_{0.588}$	$90.81_{1.272}$	$98.35_{0.161}$	$99.02_{0.074}$	$90.08_{0.309}$	$65.04_{1.179}$	$77.05_{0.631}$
OUR$_{DENI}$	$\mathbf{87.54}_{0.301}$	$91.43_{0.678}$	$\mathbf{98.56}_{0.084}$	$\mathbf{99.06}_{0.045}$	$90.57_{0.078}$	$\mathbf{66.72}_{0.275}$	$77.68_{0.409}$
OUR$_{DENIALS}$	$87.44_{0.054}$	$\mathbf{92.25}_{0.088}$	$\mathbf{98.56}_{0.068}$	$99.04_{0.045}$	$\mathbf{91.01}_{0.157}$	$66.59_{0.323}$	$\mathbf{78.24}_{0.033}$

Table 4: Comparison of the DENI method with existing mitigation strategies and baselines on full fine-tuning, LoRA, IA3 and UniPELT using BERT model. The comparison is done in terms of overall performance and deviation. The highest performance for each fine-tuning method is in **bold** and lowest deviation is underlined (not considering *All Data* baseline).

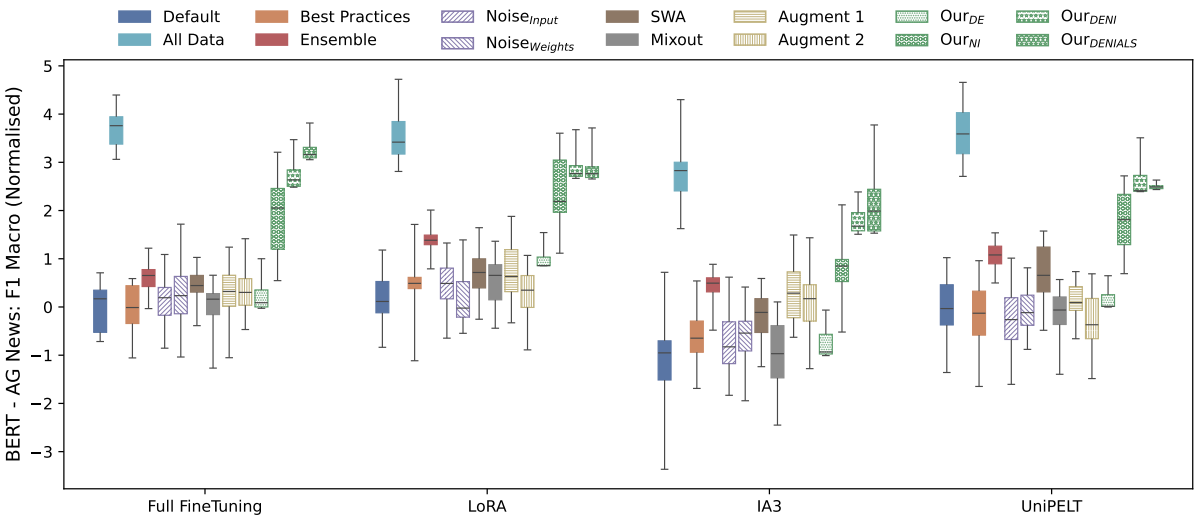

Figure 19: Benefit of mitigation strategies for the different fine-tuning methods using BERT on AG News dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

ing deviation on par or often even lower than the one from *Ensemble*. However, it also leads to lower performance than the *Ensemble* method. Only in specific cases, such as BERT on the MRPC dataset trained using UniPELT or ALBERT on the CoLA dataset trained using IA3, we observe similar performance of the *Delayed Ensemble* to the *Ensemble* or other components of the *DENI* method. The *Noisy Interpolation (NI)* leads to an increase in the performance, but often at the cost of higher standard deviation. When combined in the *DENI* method, the complementarity of these two components leads to a significant benefit, increasing the overall performance while reducing the standard deviation. Finally, combining the *DENI* method with augmented labelled samples (to obtain the *DENIALS* method) provides different benefit based on the dataset and the fine-tuning method used. In about half the cases (mainly when using full fine-tuning or UniPELT) the benefit of the *DENI* method is higher, while in other (mainly when using LoRA or IA3) the *DENIALS* method leads to higher performance and lower standard deviation. However, taking the cost of the method into consideration, the *DENI* method can be viewed as more efficient, as it requires half of the computation cost, while producing similar increase in performance and reduction in standard deviation – the difference in the performance and deviation in results between the *DENI* and *DENIALS* methods is not statistically significant (p-value of 0.14 using the Mann-Whitney U test; p-value of 0.82 using the Levene test).

To summarise, the *DENI* method provides consistent benefit across different models, datasets and fine-tuning strategies, even in cases when the deviation in results is significantly higher or even negligible. At the same time, the other mitigation strategies we compare against (with the exception of *Ensemble*) show lower consistency in their benefit. In some cases, using the mitigation strategies, especially *DENIALS* or even *DENI* leads to a performance that is approaching the one when using all data in the dataset, while also reducing the standard deviation. Finally, increasing the number of available labelled samples through augmentation has significantly higher benefit when the number of trainable parameters is low (using IA3 fine-tuning or ALBERT model), outperforming all the regular mitigation strategies that modify the optimisation process in specific cases.

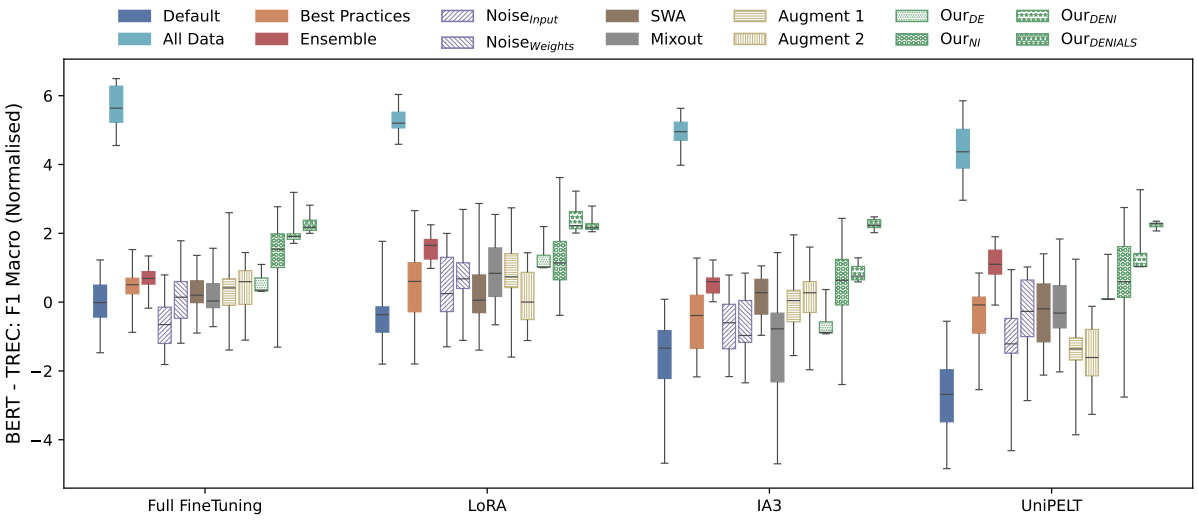

Figure 20: Benefit of mitigation strategies for the different fine-tuning methods using BERT on TREC dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

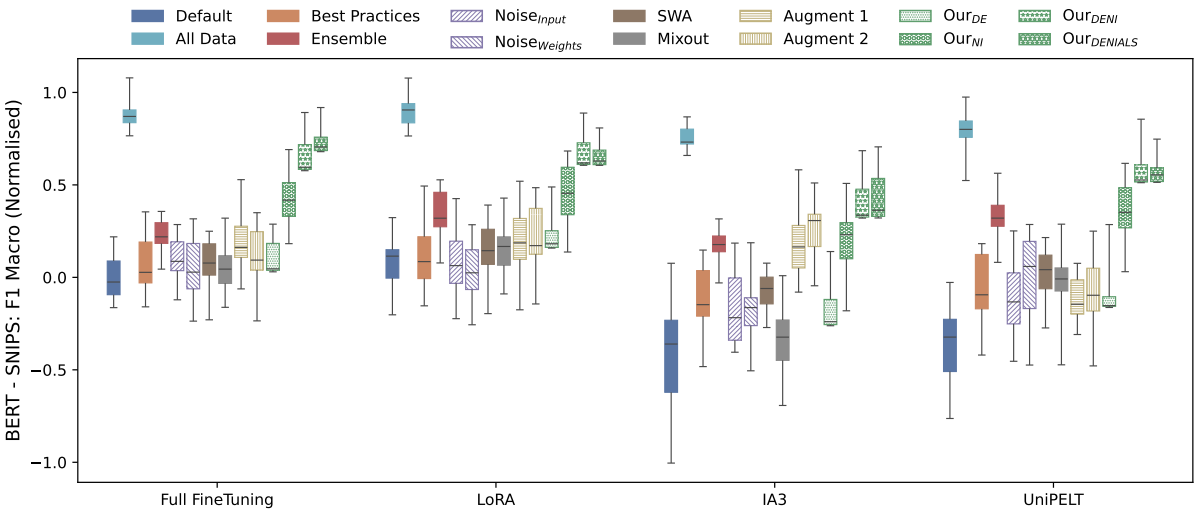

Figure 21: Benefit of mitigation strategies for the different fine-tuning methods using BERT on SNIPS dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

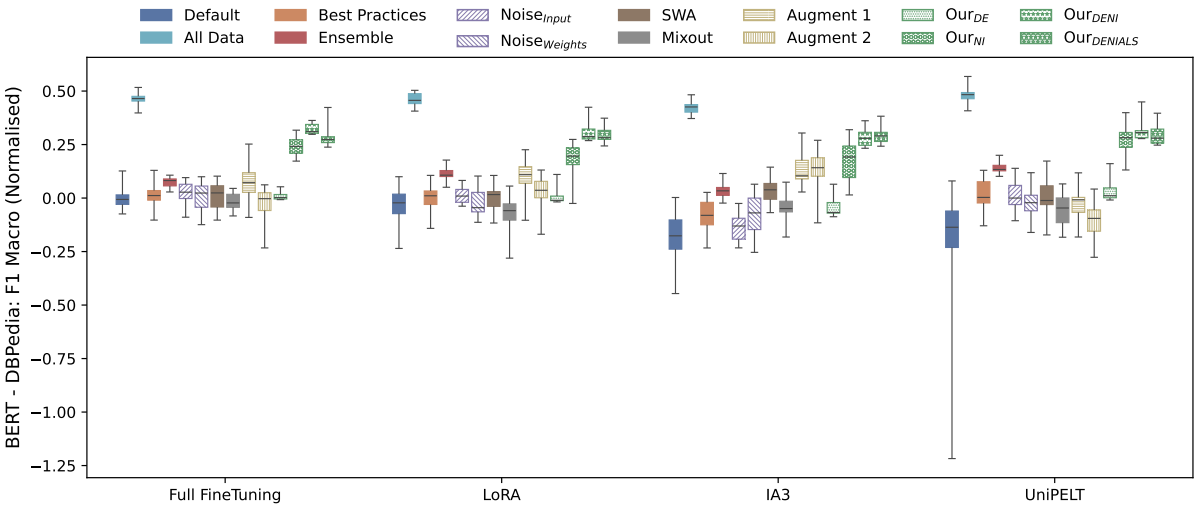

Figure 22: Benefit of mitigation strategies for the different fine-tuning methods using BERT on DBPedia dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

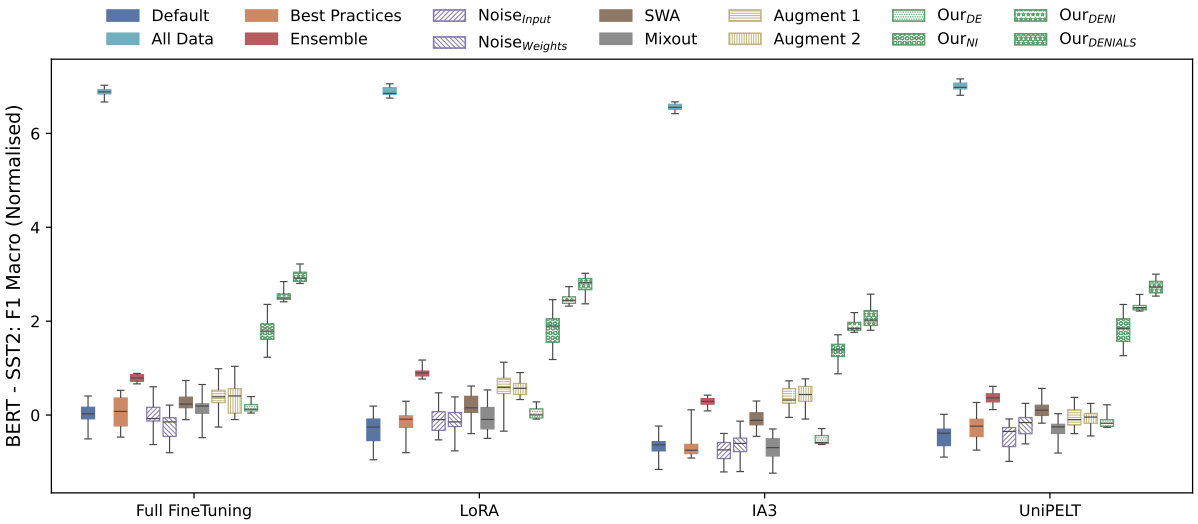

Figure 23: Benefit of mitigation strategies for the different fine-tuning methods using BERT on SST2 dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

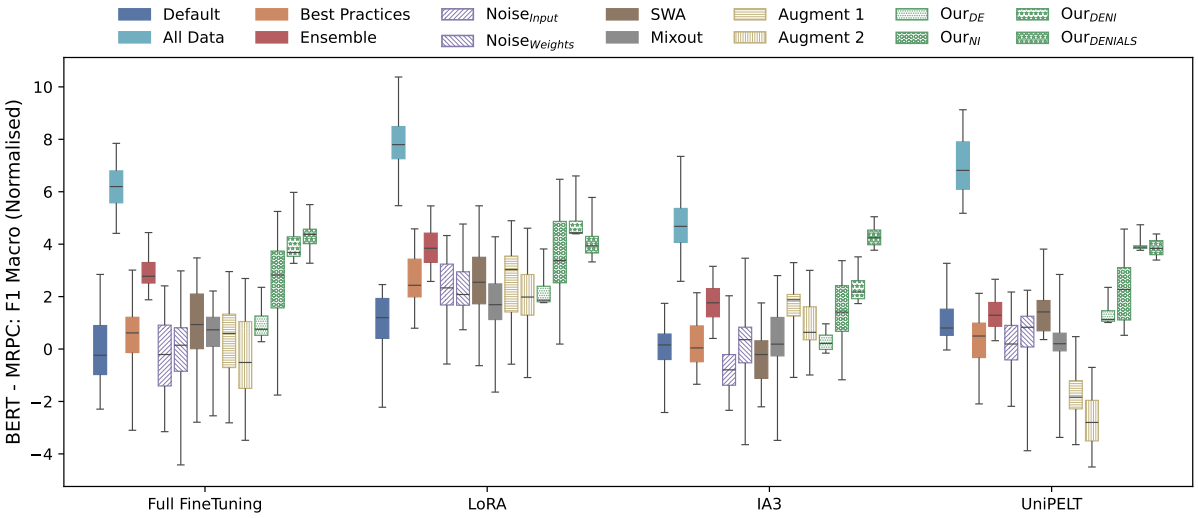

Figure 24: Benefit of mitigation strategies for the different fine-tuning methods using BERT on MRPC dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

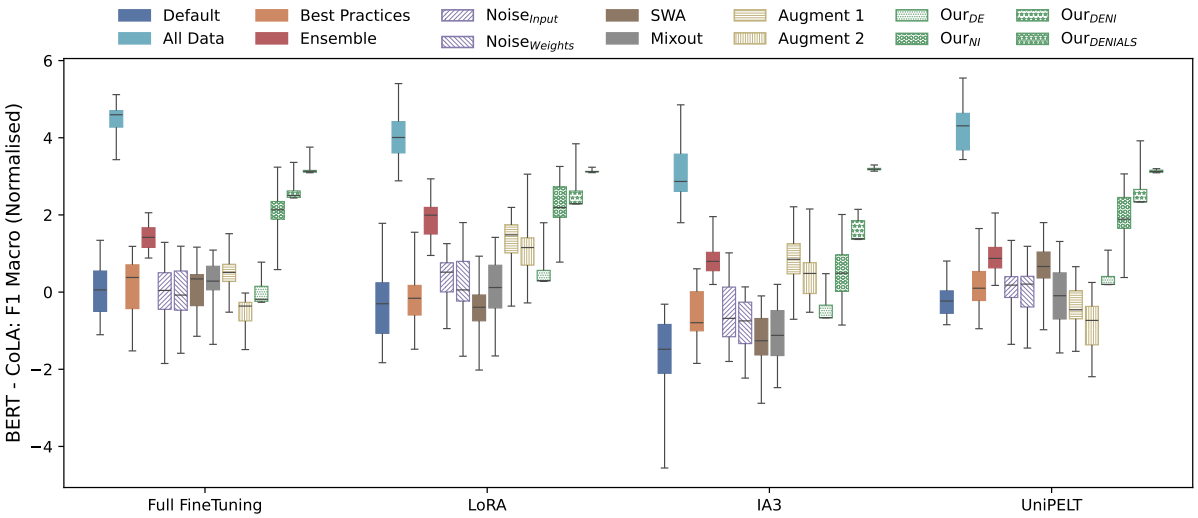

Figure 25: Benefit of mitigation strategies for the different fine-tuning methods using BERT on CoLA dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

RoBERTa	AG News	TREC	SNIPS	DBPedia	SST2	MRPC	CoLA
			Full FineTuning				
Default	$85.77_{0.433}$	$90.98_{0.979}$	$98.42_{0.092}$	$98.36_{0.082}$	$90.14_{0.270}$	$65.55_{1.654}$	$77.16_{0.934}$
All Data	$89.27_{0.391}$	$94.81_{0.691}$	$99.07_{0.086}$	$99.06_{0.028}$	$95.15_{0.084}$	$73.45_{1.031}$	$81.56_{0.632}$
Best Practices	$86.12_{0.418}$	$92.25_{0.733}$	$98.36_{0.124}$	$98.38_{0.059}$	$89.99_{0.288}$	$66.43_{1.276}$	$77.27_{0.870}$
Ensemble	$86.56_{0.236}$	$92.80_{0.378}$	$98.49_{0.073}$	$98.53_{0.023}$	$90.69_{0.119}$	$67.51_{0.703}$	$78.73_{0.431}$
Noise$_{Input}$	$86.04_{0.485}$	$92.12_{0.645}$	$98.37_{0.144}$	$98.40_{0.073}$	$89.93_{0.297}$	$66.13_{1.933}$	$77.04_{0.853}$
Noise$_{Weights}$	$86.23_{0.547}$	$91.90_{0.954}$	$98.29_{0.111}$	$98.29_{0.083}$	$89.94_{0.280}$	$66.56_{1.326}$	$76.92_{0.678}$
SWA	$86.44_{0.450}$	$91.81_{1.037}$	$98.46_{0.108}$	$98.40_{0.044}$	$90.15_{0.257}$	$66.35_{1.373}$	$76.70_{1.357}$
Mixout	$86.02_{0.471}$	$91.87_{0.724}$	$98.37_{0.126}$	$98.35_{0.059}$	$90.13_{0.251}$	$65.94_{1.641}$	$77.20_{0.760}$
Augment 1	$85.91_{0.524}$	$91.09_{0.708}$	$98.40_{0.104}$	$98.52_{0.067}$	$90.30_{0.226}$	$65.54_{1.723}$	$77.25_{0.656}$
Augment 2	$85.96_{0.413}$	$90.17_{0.770}$	$98.42_{0.103}$	$98.43_{0.068}$	$90.43_{0.246}$	$63.49_{1.031}$	$76.61_{0.631}$
Our$_{DE}$	$86.24_{0.192}$	$91.76_{0.335}$	$98.42_{0.031}$	$98.37_{0.024}$	$89.89_{0.138}$	$65.82_{0.693}$	$77.40_{0.350}$
Our$_{NI}$	$88.10_{0.530}$	$92.27_{0.972}$	$98.86_{0.105}$	$98.56_{0.061}$	$91.85_{0.375}$	$67.88_{1.692}$	$79.43_{0.882}$
Our$_{DENI}$	$\mathbf{88.67}_{0.133}$	$\mathbf{93.15}_{0.287}$	$\mathbf{98.92}_{0.027}$	$\mathbf{98.66}_{0.018}$	$92.25_{0.085}$	$\mathbf{68.23}_{0.483}$	$\mathbf{79.69}_{0.365}$
Our$_{DENIALS}$	$88.26_{0.150}$	$93.11_{0.217}$	$98.85_{0.027}$	$98.53_{0.025}$	$\mathbf{92.43}_{0.067}$	$67.83_{0.530}$	$79.31_{0.235}$
			LoRA				
Default	$85.73_{0.448}$	$89.55_{0.907}$	$98.34_{0.112}$	$98.16_{0.133}$	$89.95_{0.359}$	$64.19_{1.592}$	$76.71_{1.040}$
All Data	$89.45_{0.374}$	$94.32_{0.686}$	$98.88_{0.102}$	$99.08_{0.027}$	$95.12_{0.075}$	$72.95_{0.984}$	$81.07_{0.632}$
Best Practices	$85.87_{0.477}$	$91.12_{0.648}$	$98.25_{0.146}$	$98.25_{0.087}$	$90.13_{0.285}$	$64.21_{1.711}$	$77.25_{0.912}$
Ensemble	$86.56_{0.273}$	$92.50_{0.422}$	$98.39_{0.054}$	$98.44_{0.037}$	$90.77_{0.122}$	$65.49_{0.807}$	$78.11_{0.636}$
Noise$_{Input}$	$86.04_{0.515}$	$91.06_{1.140}$	$98.31_{0.151}$	$98.30_{0.086}$	$90.02_{0.340}$	$63.38_{2.061}$	$77.00_{0.762}$
Noise$_{Weights}$	$85.82_{0.471}$	$91.20_{0.885}$	$98.28_{0.103}$	$98.23_{0.107}$	$90.09_{0.250}$	$63.65_{2.281}$	$77.01_{0.715}$
SWA	$86.28_{0.436}$	$91.50_{0.624}$	$98.37_{0.125}$	$98.31_{0.087}$	$90.21_{0.498}$	$65.24_{2.331}$	$77.11_{0.871}$
Mixout	$85.83_{0.551}$	$91.49_{0.865}$	$98.31_{0.118}$	$98.20_{0.131}$	$90.11_{0.285}$	$64.60_{2.130}$	$77.20_{0.855}$
Augment 1	$86.36_{0.506}$	$91.19_{0.625}$	$98.49_{0.143}$	$98.32_{0.115}$	$90.60_{0.189}$	$64.38_{8.130}$	$78.11_{0.921}$
Augment 2	$86.13_{0.423}$	$90.45_{0.641}$	$98.47_{0.157}$	$98.26_{0.123}$	$90.46_{0.224}$	$64.88_{1.702}$	$77.90_{0.886}$
Our$_{DE}$	$85.61_{0.243}$	$91.10_{0.402}$	$98.29_{0.045}$	$98.21_{0.029}$	$90.10_{0.105}$	$64.52_{0.804}$	$77.26_{0.586}$
Our$_{NI}$	$87.55_{0.575}$	$92.73_{0.759}$	$98.70_{0.139}$	$98.51_{0.110}$	$91.98_{0.279}$	$66.51_{1.856}$	$79.17_{0.851}$
Our$_{DENI}$	$87.98_{0.219}$	$93.01_{0.226}$	$98.84_{0.029}$	$98.58_{0.027}$	$92.44_{0.072}$	$66.95_{0.469}$	$79.74_{0.397}$
Our$_{DENIALS}$	$\mathbf{88.14}_{0.175}$	$\mathbf{93.25}_{0.205}$	$\mathbf{98.96}_{0.038}$	$\mathbf{98.67}_{0.034}$	$\mathbf{92.70}_{0.070}$	$\mathbf{67.06}_{0.490}$	$\mathbf{79.95}_{0.882}$
			IA3				
Default	$84.98_{0.557}$	$90.59_{0.887}$	$98.07_{0.156}$	$97.89_{0.128}$	$89.00_{0.230}$	$58.58_{4.528}$	$73.88_{2.586}$
All Data	$88.83_{0.562}$	$94.66_{0.560}$	$98.93_{0.094}$	$98.85_{0.059}$	$94.51_{0.122}$	$64.83_{8.397}$	$73.03_{13.417}$
Best Practices	$85.03_{0.503}$	$86.20_{18.439}$	$93.32_{20.601}$	$97.69_{0.192}$	$88.51_{0.374}$	$55.34_{8.258}$	$73.04_{7.342}$
Ensemble	$86.39_{0.296}$	$91.49_{0.441}$	$98.65_{0.072}$	$98.29_{0.063}$	$90.11_{0.158}$	$57.73_{1.880}$	$76.63_{0.542}$
Noise$_{Input}$	$84.81_{0.557}$	$90.35_{1.104}$	$98.13_{0.199}$	$97.63_{0.192}$	$88.46_{0.395}$	$56.39_{5.678}$	$71.41_{10.636}$
Noise$_{Weights}$	$80.88_{15.803}$	$90.29_{1.211}$	$98.16_{0.239}$	$97.61_{0.179}$	$87.97_{0.409}$	$49.52_{13.034}$	$71.59_{7.086}$
SWA	$86.13_{0.474}$	$89.66_{0.951}$	$98.29_{0.175}$	$95.47_{10.860}$	$88.02_{0.426}$	$52.74_{12.001}$	$68.29_{6.409}$
Mixout	$84.81_{0.507}$	$90.26_{1.052}$	$98.02_{0.215}$	$97.97_{0.119}$	$85.22_{11.347}$	$33.22_{7.739}$	$33.01_{9.143}$
Augment 1	$86.30_{0.660}$	$91.48_{0.846}$	$98.39_{0.192}$	$98.21_{0.146}$	$86.81_{12.490}$	$56.49_{4.422}$	$75.56_{6.946}$
Augment 2	$86.41_{0.543}$	$91.19_{0.796}$	$98.64_{0.176}$	$98.26_{0.129}$	$90.21_{0.261}$	$57.95_{1.282}$	$\mathbf{78.70}_{0.790}$
Our$_{DE}$	$84.93_{0.275}$	$89.83_{0.495}$	$98.19_{0.053}$	$97.73_{0.058}$	$88.55_{0.107}$	$58.96_{0.407}$	$75.09_{0.396}$
Our$_{NI}$	$87.49_{0.673}$	$91.65_{0.952}$	$98.64_{0.163}$	$98.43_{0.196}$	$90.33_{0.313}$	$58.43_{10.157}$	$74.31_{11.236}$
Our$_{DENI}$	$87.70_{0.225}$	$91.85_{0.457}$	$98.79_{0.035}$	$98.56_{0.037}$	$90.82_{0.108}$	$62.40_{0.517}$	$77.25_{0.559}$
Our$_{DENIALS}$	$\mathbf{87.75}_{0.207}$	$\mathbf{92.00}_{0.337}$	$\mathbf{98.88}_{0.042}$	$\mathbf{98.61}_{0.042}$	$\mathbf{91.02}_{0.116}$	$\mathbf{62.81}_{0.554}$	$77.41_{0.546}$
			UniPELT				
Default	$85.98_{0.359}$	$91.84_{0.870}$	$98.36_{0.108}$	$98.42_{0.066}$	$90.34_{0.304}$	$62.43_{7.489}$	$76.30_{8.074}$
All Data	$89.73_{0.353}$	$94.31_{0.526}$	$99.02_{0.114}$	$99.08_{0.039}$	$95.07_{0.093}$	$73.33_{0.873}$	$82.03_{0.407}$
Best Practices	$86.07_{0.446}$	$92.62_{0.777}$	$98.35_{0.121}$	$98.38_{0.102}$	$90.26_{0.252}$	$64.92_{1.433}$	$78.45_{0.754}$
Ensemble	$87.18_{0.240}$	$93.25_{0.575}$	$98.59_{0.076}$	$98.55_{0.024}$	$91.00_{0.067}$	$66.04_{0.656}$	$80.26_{0.372}$
Noise$_{Input}$	$86.16_{0.452}$	$92.18_{0.809}$	$98.42_{0.137}$	$98.38_{0.098}$	$90.25_{0.206}$	$59.90_{10.928}$	$76.04_{10.489}$
Noise$_{Weights}$	$85.96_{0.623}$	$92.41_{0.745}$	$98.32_{0.123}$	$98.25_{0.114}$	$90.31_{0.198}$	$65.50_{1.254}$	$78.07_{0.758}$
SWA	$86.64_{0.402}$	$92.28_{0.822}$	$98.40_{0.133}$	$98.42_{0.088}$	$88.62_{8.224}$	$62.66_{7.641}$	$70.06_{19.817}$
Mixout	$86.27_{0.550}$	$92.23_{1.027}$	$98.36_{0.163}$	$98.42_{0.079}$	$90.22_{0.257}$	$64.66_{1.695}$	$66.15_{21.219}$
Augment 1	$86.02_{0.432}$	$91.88_{1.072}$	$98.35_{0.136}$	$98.55_{0.073}$	$90.42_{0.252}$	$60.40_{1.399}$	$75.97_{11.790}$
Augment 2	$85.89_{0.459}$	$90.88_{0.564}$	$98.33_{0.153}$	$98.53_{0.089}$	$90.18_{0.230}$	$59.23_{1.251}$	$78.48_{0.800}$
Our$_{DE}$	$86.09_{0.235}$	$92.39_{0.417}$	$98.36_{0.056}$	$98.42_{0.026}$	$90.27_{0.106}$	$65.42_{0.512}$	$79.77_{0.445}$
Our$_{NI}$	$87.95_{0.493}$	$92.99_{1.076}$	$98.76_{0.144}$	$98.56_{0.110}$	$92.17_{0.262}$	$65.97_{5.566}$	$72.04_{19.807}$
Our$_{DENI}$	$\mathbf{88.56}_{0.189}$	$\mathbf{93.68}_{0.284}$	$98.91_{0.043}$	$\mathbf{98.68}_{0.018}$	$92.67_{0.072}$	$\mathbf{67.56}_{0.275}$	$\mathbf{80.68}_{0.343}$
Our$_{DENIALS}$	$\mathbf{88.56}_{0.189}$	$\mathbf{93.68}_{0.298}$	$\mathbf{98.94}_{0.037}$	$98.67_{0.030}$	$\mathbf{92.70}_{0.084}$	$67.46_{0.396}$	$80.55_{0.241}$

Table 5: Comparison of the DENI method with existing mitigation strategies and baselines on full fine-tuning, LoRA, IA3 and UniPELT using RoBERTa model. The comparison is done in terms of overall performance and deviation. The highest performance for each fine-tuning method is in **bold** and lowest deviation is underlined (not considering *All Data* baseline).

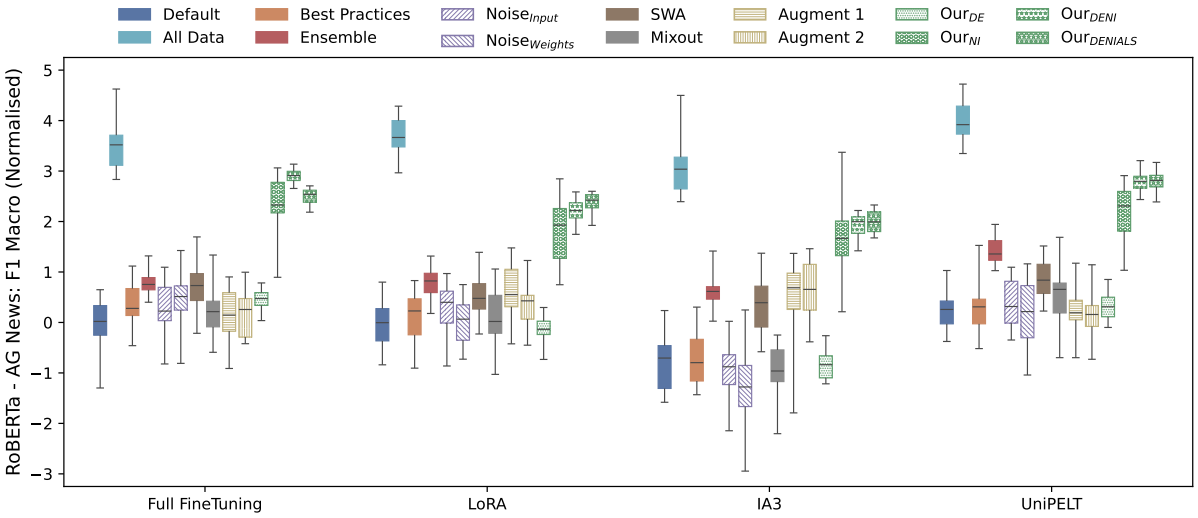

Figure 26: Benefit of mitigation strategies for the different fine-tuning methods using RoBERTa on AG News dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

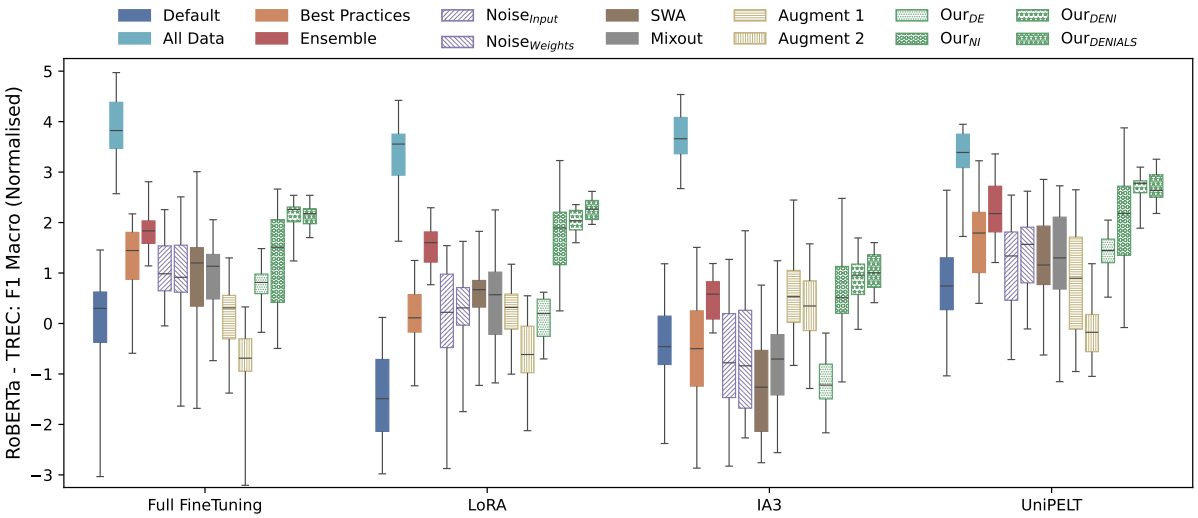

Figure 27: Benefit of mitigation strategies for the different fine-tuning methods using RoBERTa on TREC dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

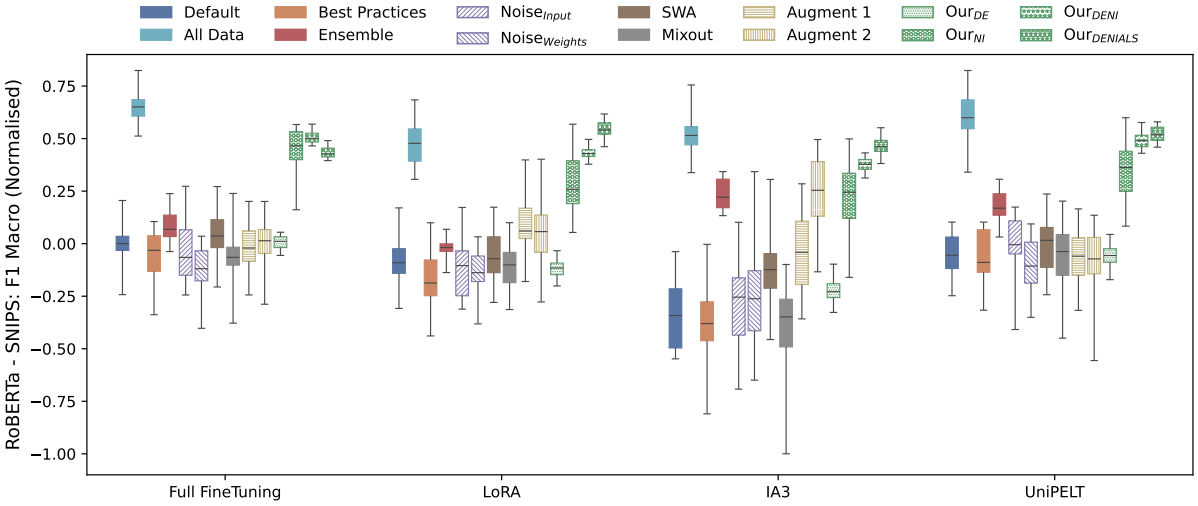

Figure 28: Benefit of mitigation strategies for the different fine-tuning methods using RoBERTa on SNIPS dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

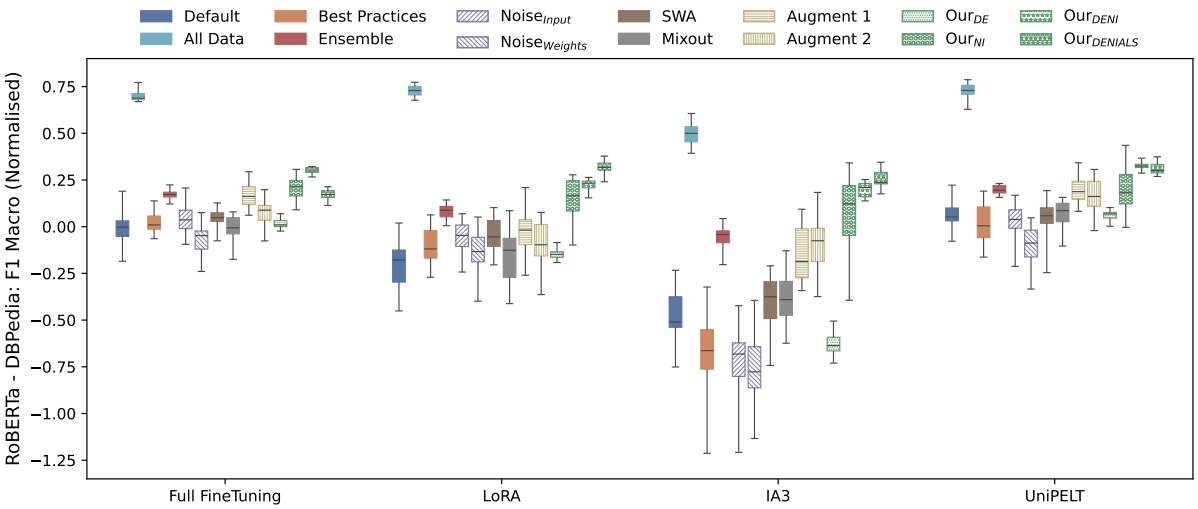

Figure 29: Benefit of mitigation strategies for the different fine-tuning methods using RoBERTa on DBPedia dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

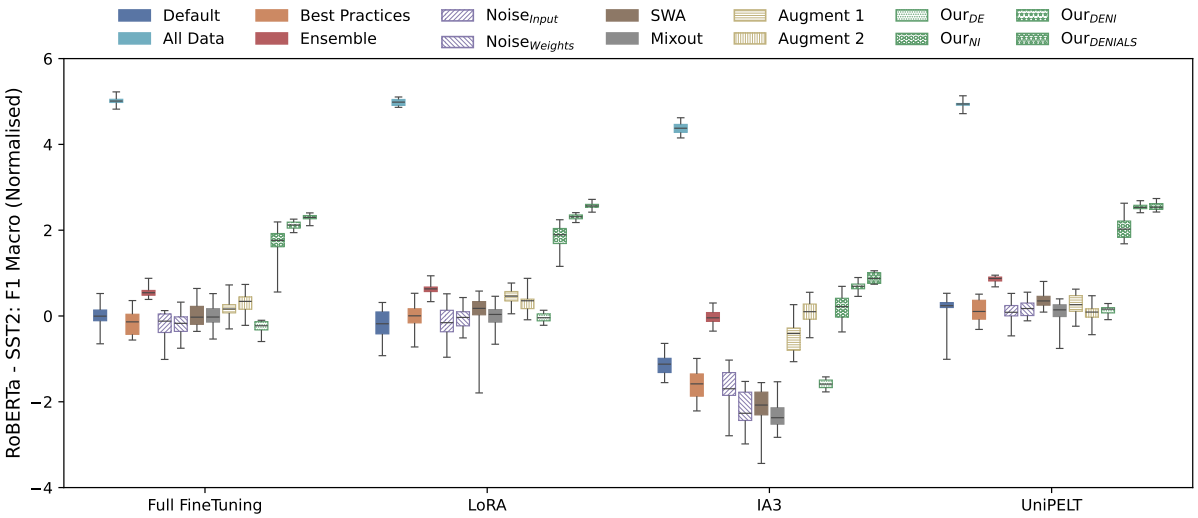

Figure 30: Benefit of mitigation strategies for the different fine-tuning methods using RoBERTa on SST2 dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

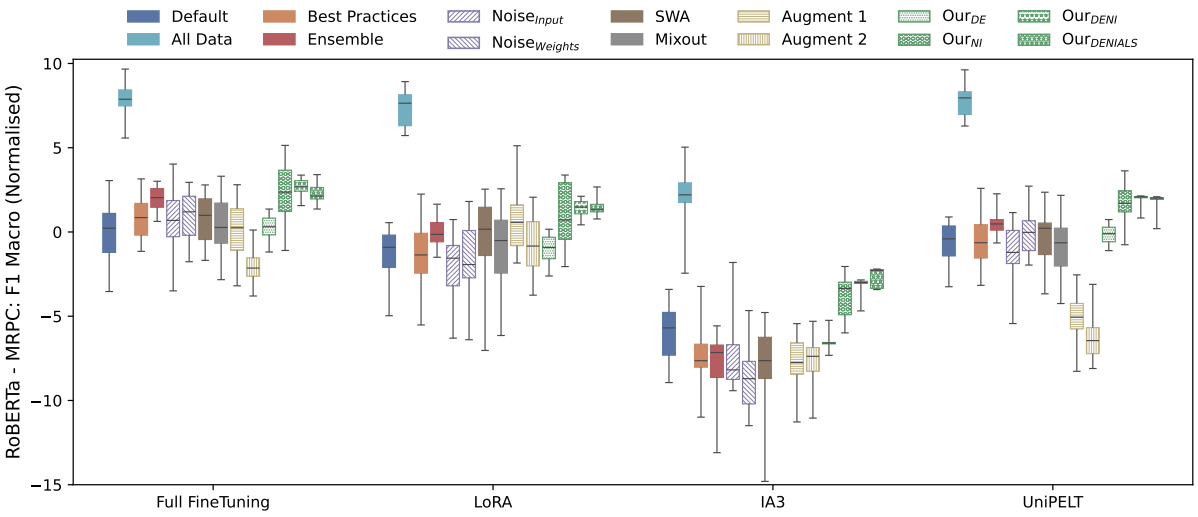

Figure 31: Benefit of mitigation strategies for the different fine-tuning methods using RoBERTa on MRPC dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

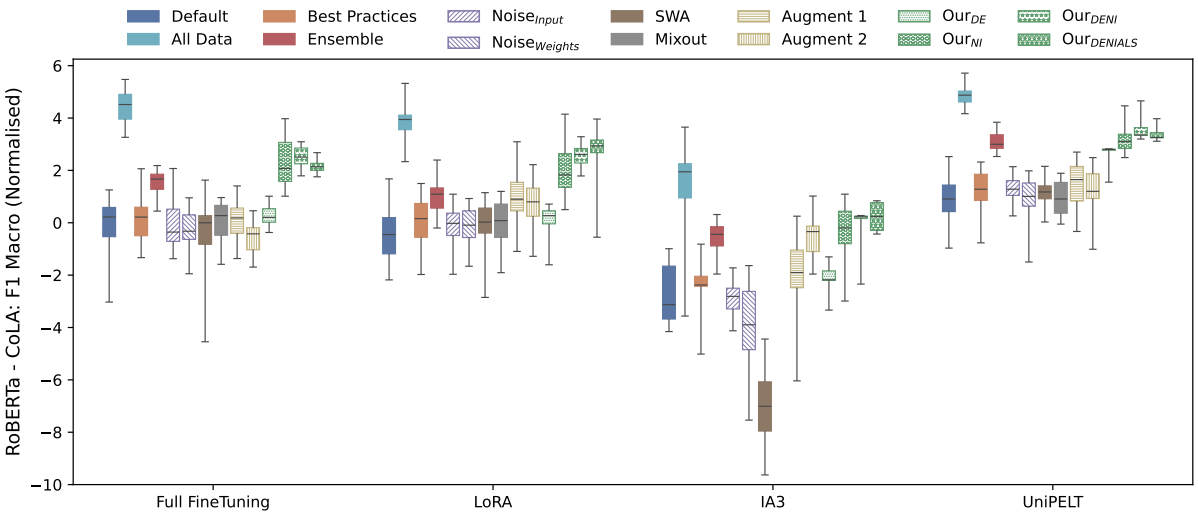

Figure 32: Benefit of mitigation strategies for the different fine-tuning methods using RoBERTa on CoLA dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

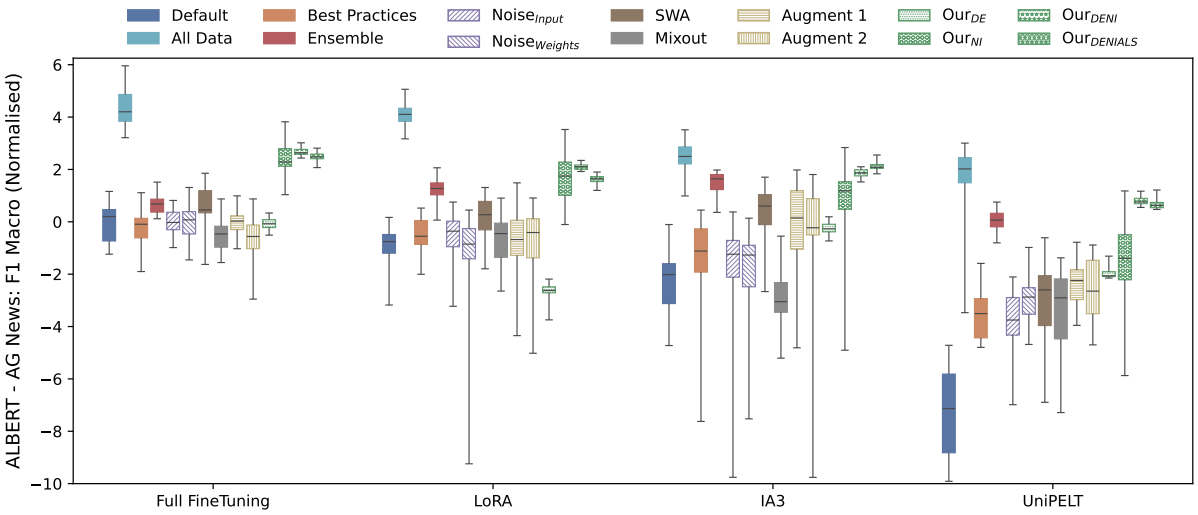

Figure 33: Benefit of mitigation strategies for the different fine-tuning methods using ALBERT on AG News dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

ALBERT	AG NEWS	TREC	SNIPS	DBPEDIA	SST2	MRPC	COLA
			FULL FINETUNING				
DEFAULT	$83.11_{0.716}$	$88.45_{1.794}$	$97.23_{0.245}$	$98.23_{0.127}$	$85.49_{3.983}$	$60.12_{2.438}$	$65.41_{9.853}$
ALL DATA	$87.46_{0.680}$	$93.98_{0.912}$	$98.62_{0.126}$	$99.05_{0.039}$	$94.58_{0.189}$	$67.07_{7.899}$	$78.94_{1.257}$
BEST PRACTICES	$82.95_{0.679}$	$89.66_{1.033}$	$97.30_{0.222}$	$98.21_{0.143}$	$86.03_{1.169}$	$60.20_{3.691}$	$70.22_{1.473}$
ENSEMBLE	$83.76_{0.347}$	$90.67_{0.376}$	$97.83_{0.122}$	$98.44_{0.036}$	$88.20_{0.151}$	$66.41_{1.571}$	$72.13_{0.664}$
NOISE$_{Input}$	$83.07_{0.489}$	$88.33_{3.974}$	$97.27_{0.269}$	$98.13_{0.320}$	$84.64_{7.497}$	$60.63_{4.068}$	$68.93_{2.667}$
NOISE$_{Weights}$	$83.09_{0.683}$	$89.24_{1.021}$	$97.19_{0.349}$	$97.82_{0.312}$	$85.40_{2.145}$	$60.46_{2.962}$	$69.54_{1.370}$
SWA	$83.77_{0.766}$	$89.18_{1.447}$	$97.49_{0.264}$	$98.07_{0.367}$	$86.06_{2.042}$	$61.05_{3.537}$	$67.06_{8.933}$
MIXOUT	$82.64_{0.688}$	$89.43_{1.237}$	$97.37_{0.321}$	$98.17_{0.177}$	$86.09_{0.901}$	$59.62_{4.338}$	$67.56_{7.110}$
AUGMENT 1	$83.09_{0.522}$	$89.66_{1.202}$	$97.28_{0.329}$	$98.23_{0.205}$	$86.46_{1.782}$	$60.80_{4.930}$	$69.96_{3.846}$
AUGMENT 2	$82.47_{0.820}$	$88.69_{1.292}$	$97.32_{0.283}$	$98.18_{0.118}$	$87.05_{0.823}$	$60.55_{2.952}$	$70.36_{3.699}$
OUR$_{DE}$	$83.02_{0.209}$	$89.21_{0.346}$	$97.34_{0.079}$	$98.09_{0.044}$	$86.26_{0.225}$	$60.26_{0.919}$	$70.17_{0.568}$
OUR$_{NI}$	$85.50_{0.652}$	$90.76_{0.907}$	$97.95_{0.667}$	$98.46_{0.245}$	$88.89_{0.603}$	$66.70_{2.416}$	$72.84_{1.095}$
OUR$_{DENI}$	$\mathbf{85.78}_{0.156}$	$\mathbf{91.00}_{0.241}$	$\mathbf{98.21}_{0.166}$	$\mathbf{98.68}_{0.055}$	$\mathbf{89.49}_{0.176}$	$\mathbf{67.51}_{0.628}$	$\mathbf{73.09}_{0.441}$
OUR$_{DENIALS}$	$85.58_{0.169}$	$90.86_{0.331}$	$97.95_{0.067}$	$98.39_{0.052}$	$89.37_{0.102}$	$67.40_{0.577}$	$72.61_{0.437}$
			LORA				
DEFAULT	$79.77_{10.530}$	$76.57_{23.043}$	$87.12_{27.864}$	$92.62_{21.163}$	$72.73_{21.716}$	$48.45_{14.781}$	$58.72_{14.973}$
ALL DATA	$87.17_{0.451}$	$81.71_{22.575}$	$91.14_{22.510}$	$98.93_{0.506}$	$87.56_{17.387}$	$62.28_{12.957}$	$59.86_{18.586}$
BEST PRACTICES	$78.77_{16.671}$	$82.90_{18.145}$	$96.15_{3.883}$	$97.89_{0.206}$	$75.02_{18.841}$	$61.43_{5.798}$	$55.87_{18.034}$
ENSEMBLE	$84.36_{0.452}$	$90.11_{0.699}$	$97.86_{0.133}$	$98.44_{0.052}$	$86.67_{2.238}$	$64.67_{4.205}$	$69.33_{10.165}$
NOISE$_{Input}$	$82.63_{0.866}$	$76.36_{25.050}$	$87.66_{28.042}$	$97.92_{0.188}$	$74.27_{21.892}$	$55.94_{14.109}$	$61.62_{18.108}$
NOISE$_{Weights}$	$81.90_{2.025}$	$83.50_{9.570}$	$88.98_{22.796}$	$86.63_{27.099}$	$74.20_{21.399}$	$57.76_{11.426}$	$57.38_{17.570}$
SWA	$74.06_{23.224}$	$69.66_{32.682}$	$83.42_{33.489}$	$87.86_{26.928}$	$54.70_{23.038}$	$48.28_{14.489}$	$48.01_{17.929}$
MIXOUT	$82.45_{0.905}$	$83.40_{18.324}$	$97.00_{0.983}$	$96.83_{4.359}$	$74.78_{21.506}$	$56.87_{12.139}$	$58.36_{19.808}$
AUGMENT 1	$82.33_{1.315}$	$76.75_{24.220}$	$87.80_{27.988}$	$97.95_{1.020}$	$72.68_{22.844}$	$55.59_{11.736}$	$58.15_{17.500}$
AUGMENT 2	$82.37_{1.293}$	$78.13_{24.119}$	$92.70_{20.126}$	$98.09_{0.163}$	$76.13_{20.011}$	$58.94_{9.478}$	$64.62_{13.715}$
OUR$_{DE}$	$80.47_{0.316}$	$88.09_{0.689}$	$97.83_{0.123}$	$98.24_{0.074}$	$86.18_{0.475}$	$64.84_{0.956}$	$71.67_{1.595}$
OUR$_{NI}$	$84.78_{0.901}$	$88.20_{5.218}$	$97.37_{1.502}$	$93.41_{21.096}$	$84.35_{11.181}$	$60.77_{11.306}$	$59.19_{21.022}$
OUR$_{DENI}$	$\mathbf{85.21}_{0.111}$	$\mathbf{91.82}_{0.560}$	$\mathbf{98.45}_{0.079}$	$98.79_{0.047}$	$87.92_{0.774}$	$\mathbf{68.87}_{0.943}$	$72.95_{1.791}$
OUR$_{DENIALS}$	$84.73_{0.149}$	$91.76_{0.598}$	$98.02_{0.504}$	$\mathbf{98.81}_{0.032}$	$\mathbf{87.93}_{1.055}$	$68.76_{0.721}$	$\mathbf{73.36}_{1.730}$
			IA3				
DEFAULT	$80.90_{1.296}$	$77.85_{11.902}$	$96.34_{0.702}$	$95.99_{5.347}$	$73.63_{17.822}$	$51.37_{10.640}$	$54.89_{13.130}$
ALL DATA	$85.57_{0.631}$	$87.23_{17.164}$	$97.97_{1.760}$	$98.51_{0.818}$	$90.37_{2.450}$	$65.53_{3.442}$	$63.60_{13.290}$
BEST PRACTICES	$78.84_{7.546}$	$84.22_{5.928}$	$96.72_{0.469}$	$95.04_{9.257}$	$77.77_{12.933}$	$55.46_{4.093}$	$50.14_{11.646}$
ENSEMBLE	$84.58_{0.441}$	$90.11_{0.830}$	$97.73_{0.118}$	$98.19_{0.064}$	$84.99_{1.262}$	$56.57_{6.732}$	$54.84_{11.430}$
NOISE$_{Input}$	$74.27_{21.131}$	$76.73_{24.718}$	$92.85_{13.980}$	$93.18_{18.298}$	$75.14_{14.414}$	$51.77_{11.849}$	$47.77_{15.243}$
NOISE$_{Weights}$	$81.05_{2.024}$	$77.91_{20.630}$	$95.05_{5.857}$	$95.66_{4.713}$	$75.60_{11.375}$	$54.77_{7.076}$	$48.41_{9.886}$
SWA	$81.37_{6.187}$	$85.44_{11.288}$	$93.33_{15.927}$	$96.12_{6.814}$	$73.04_{18.684}$	$51.83_{12.638}$	$54.21_{15.483}$
MIXOUT	$78.54_{5.491}$	$56.34_{28.193}$	$84.67_{23.558}$	$86.39_{22.527}$	$46.27_{17.586}$	$42.62_{13.977}$	$36.26_{9.978}$
AUGMENT 1	$82.76_{1.926}$	$82.25_{13.731}$	$95.11_{4.686}$	$97.30_{3.050}$	$80.76_{11.785}$	$54.16_{9.843}$	$64.78_{7.860}$
AUGMENT 2	$82.39_{3.087}$	$79.28_{18.981}$	$93.06_{19.200}$	$97.93_{0.854}$	$81.69_{10.545}$	$58.18_{4.052}$	$\mathbf{67.19}_{6.209}$
OUR$_{DE}$	$82.87_{0.209}$	$87.88_{0.407}$	$97.66_{0.075}$	$97.94_{0.097}$	$84.84_{0.698}$	$59.99_{0.408}$	$61.99_{1.657}$
OUR$_{NI}$	$83.82_{1.657}$	$88.99_{2.079}$	$97.64_{0.225}$	$98.18_{0.366}$	$81.50_{12.484}$	$60.40_{4.143}$	$57.08_{15.157}$
OUR$_{DENI}$	$84.96_{0.143}$	$\mathbf{91.02}_{0.320}$	$97.94_{0.066}$	$98.35_{0.051}$	$86.88_{0.644}$	$\mathbf{62.55}_{0.468}$	$63.35_{2.045}$
OUR$_{DENIALS}$	$\mathbf{85.24}_{0.163}$	$90.99_{0.716}$	$\mathbf{98.08}_{0.123}$	$\mathbf{98.53}_{0.275}$	$\mathbf{87.27}_{0.441}$	$62.51_{0.751}$	$64.12_{2.093}$
			UNIPELT				
DEFAULT	$74.93_{2.312}$	$51.52_{6.672}$	$83.82_{7.394}$	$83.05_{16.371}$	$80.73_{8.096}$	$58.91_{1.770}$	$60.37_{6.526}$
ALL DATA	$84.21_{3.139}$	$78.14_{4.317}$	$97.82_{0.441}$	$99.00_{0.034}$	$93.10_{2.212}$	$65.22_{1.804}$	$74.23_{7.717}$
BEST PRACTICES	$79.66_{0.960}$	$69.89_{8.061}$	$93.48_{2.481}$	$95.52_{1.320}$	$84.04_{1.959}$	$57.31_{2.411}$	$63.67_{7.417}$
ENSEMBLE	$83.15_{0.387}$	$84.49_{1.287}$	$96.92_{0.308}$	$97.61_{0.161}$	$87.17_{0.487}$	$60.63_{0.955}$	$68.46_{1.455}$
NOISE$_{Input}$	$76.92_{8.502}$	$66.41_{7.107}$	$92.94_{2.395}$	$89.56_{20.237}$	$81.47_{11.838}$	$57.85_{2.175}$	$65.19_{4.367}$
NOISE$_{Weights}$	$80.16_{0.903}$	$73.46_{7.650}$	$94.69_{1.304}$	$92.64_{14.449}$	$82.37_{8.680}$	$59.22_{2.550}$	$61.56_{11.011}$
SWA	$79.99_{1.582}$	$74.04_{6.722}$	$94.51_{1.184}$	$96.05_{1.194}$	$81.97_{9.083}$	$58.29_{2.459}$	$60.22_{14.255}$
MIXOUT	$79.63_{1.743}$	$72.86_{5.823}$	$94.55_{1.471}$	$94.38_{4.587}$	$79.05_{12.674}$	$57.85_{1.957}$	$63.49_{6.818}$
AUGMENT 1	$77.67_{13.450}$	$82.67_{2.436}$	$95.55_{0.665}$	$97.26_{0.547}$	$82.55_{9.967}$	$57.88_{1.454}$	$68.52_{3.480}$
AUGMENT 2	$80.47_{1.248}$	$81.60_{2.290}$	$95.36_{0.597}$	$97.08_{0.884}$	$83.69_{6.801}$	$57.47_{1.482}$	$68.55_{1.417}$
OUR$_{DE}$	$81.15_{0.217}$	$80.47_{1.109}$	$96.25_{0.325}$	$97.23_{0.096}$	$85.81_{0.349}$	$59.88_{0.489}$	$68.23_{0.794}$
OUR$_{NI}$	$81.01_{3.406}$	$76.86_{5.754}$	$95.47_{1.276}$	$97.40_{0.670}$	$85.38_{3.221}$	$59.73_{1.891}$	$64.91_{6.295}$
OUR$_{DENI}$	$\mathbf{83.91}_{0.160}$	$85.48_{0.914}$	$\mathbf{97.46}_{0.221}$	$97.89_{0.095}$	$\mathbf{87.89}_{0.335}$	$62.32_{0.309}$	$72.07_{0.552}$
OUR$_{DENIALS}$	$83.78_{0.182}$	$\mathbf{86.20}_{0.529}$	$96.74_{0.090}$	$\mathbf{98.96}_{0.047}$	$87.64_{0.564}$	$\mathbf{62.79}_{0.232}$	$\mathbf{72.60}_{0.231}$

Table 6: Comparison of the DENI method with existing mitigation strategies and baselines on full fine-tuning, LoRA, IA3 and UniPELT using ALBERT model. The comparison is done in terms of overall performance and deviation. The highest performance for each fine-tuning method is in **bold** and lowest deviation is underlined (not considering *All Data* baseline).

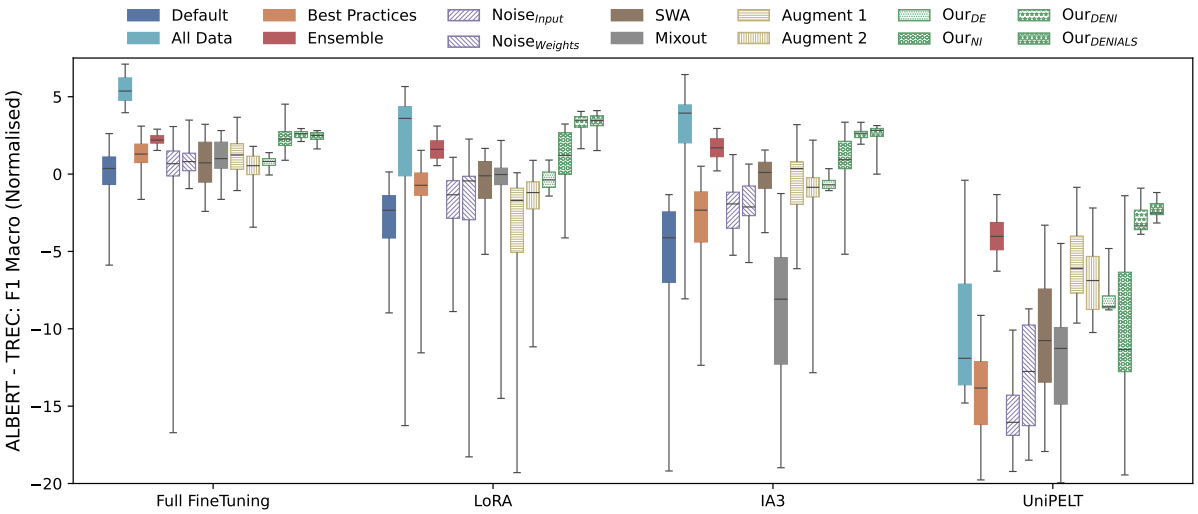

Figure 34: Benefit of mitigation strategies for the different fine-tuning methods using ALBERT on TREC dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

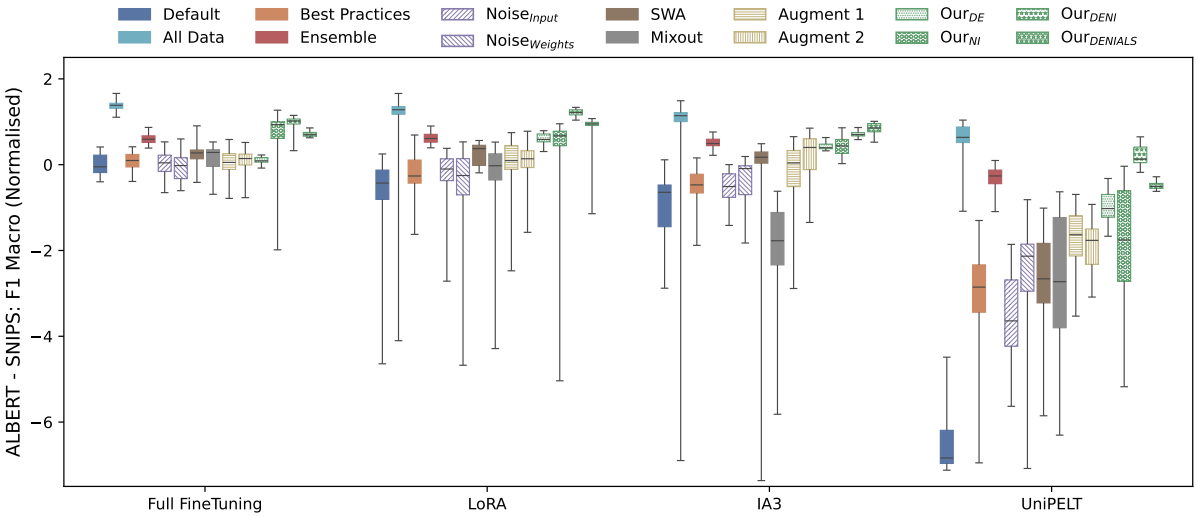

Figure 35: Benefit of mitigation strategies for the different fine-tuning methods using ALBERT on SNIPS dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

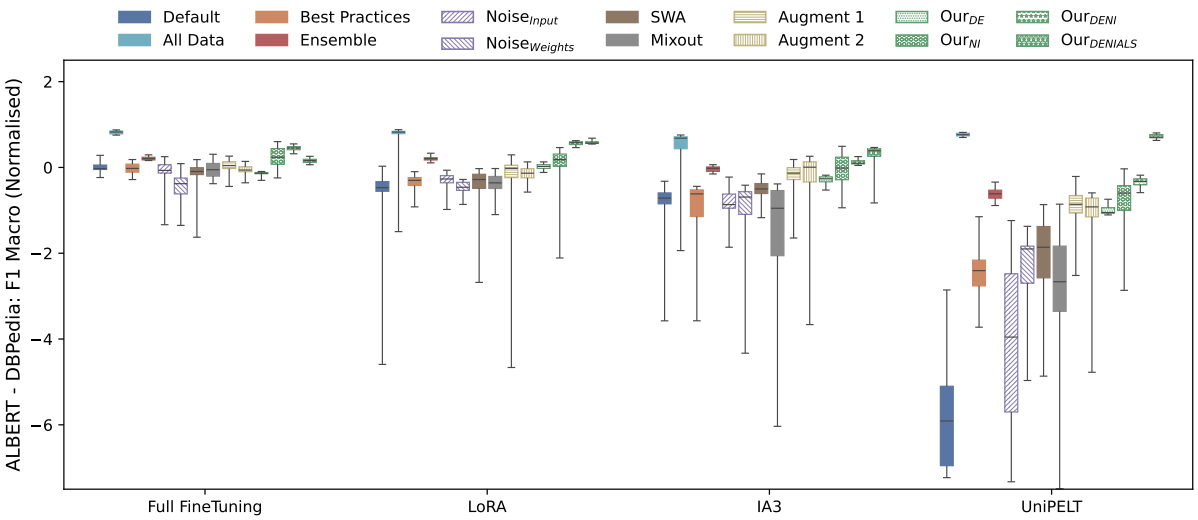

Figure 36: Benefit of mitigation strategies for the different fine-tuning methods using ALBERT on DBPedia dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

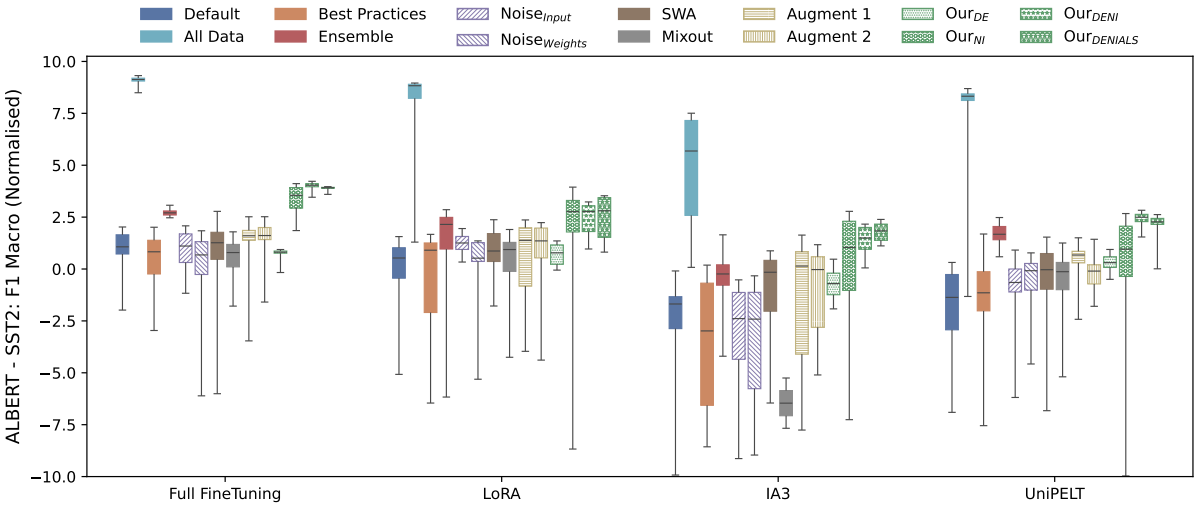

Figure 37: Benefit of mitigation strategies for the different fine-tuning methods using ALBERT on SST2 dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

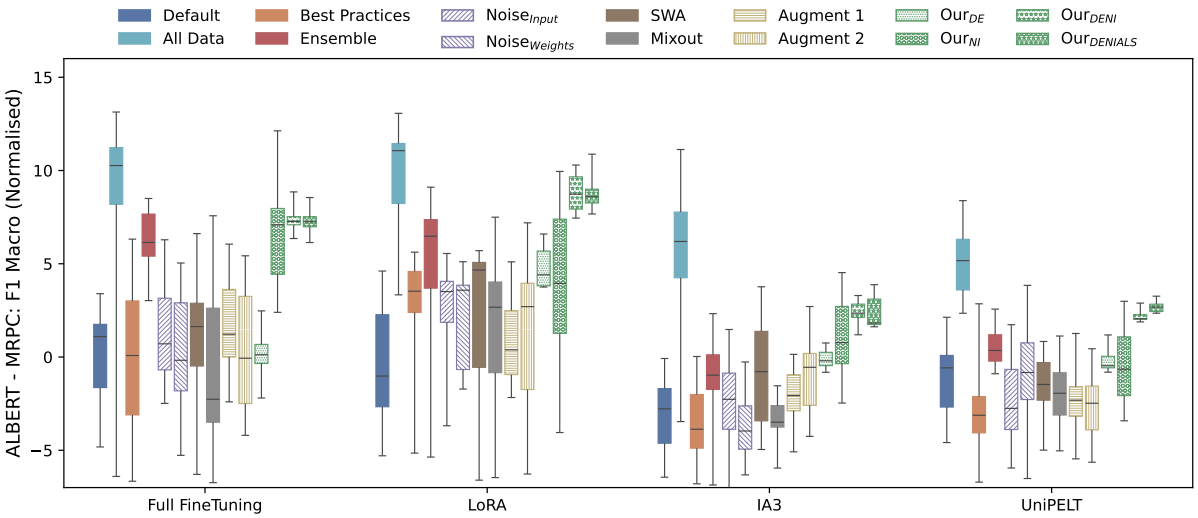

Figure 38: Benefit of mitigation strategies for the different fine-tuning methods using ALBERT on MRPC dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).

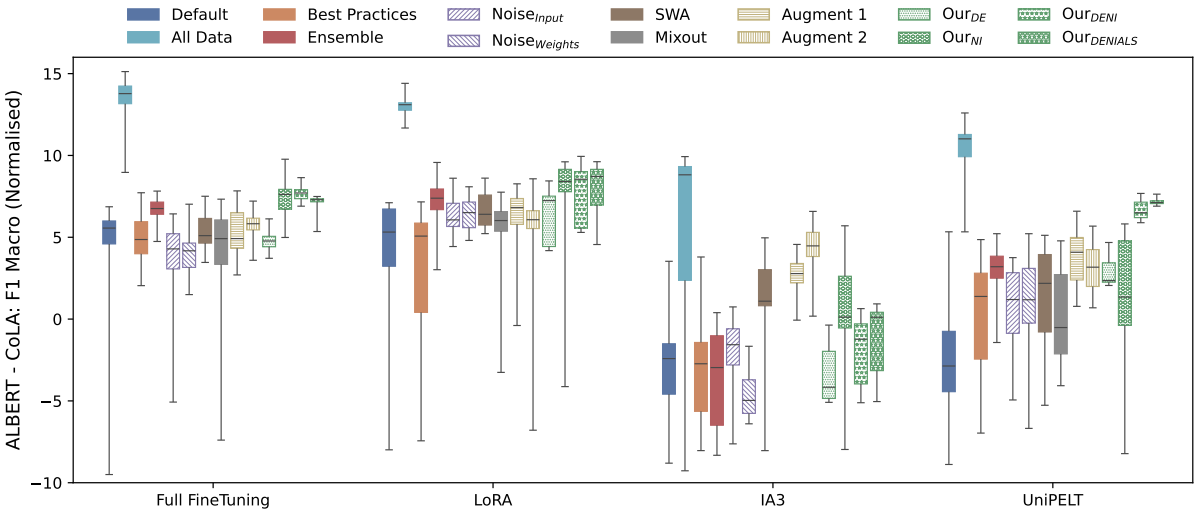

Figure 39: Benefit of mitigation strategies for the different fine-tuning methods using ALBERT on CoLA dataset. The benefit is calculated as difference to the mean performance of the *Default* baseline when using full fine-tuning. The different mitigation strategies are beneficial for all fine-tuning methods, but with different overall benefit (e.g., *Augment* on IA3).