Extreme Miscalibration and the Illusion of Adversarial Robustness

Vyas Raina[*†] **Samson Tan**[*‡] **Volkan Cevher**[‡,¶]
Aditya Rawal[‡] **Sheng Zha**[‡] **George Karypis**[‡]

[†]University of Cambridge [‡]Amazon
[¶]LIONS, IEM, STI, Ecole Polytechnique Federale de Lausanne
vr313@cam.ac.uk samson@amazon.com

Abstract

Deep learning-based Natural Language Processing (NLP) models are vulnerable to adversarial attacks, where small perturbations can cause a model to misclassify. Adversarial Training (AT) is often used to increase model robustness. However, we have discovered an intriguing phenomenon: deliberately or accidentally miscalibrating models masks gradients in a way that interferes with adversarial attack search methods, giving rise to an apparent increase in robustness. We show that this observed gain in robustness is an illusion of robustness (IOR), and demonstrate how an adversary can perform various forms of test-time temperature calibration to nullify the aforementioned interference and allow the adversarial attack to find adversarial examples. Hence, we urge the NLP community to incorporate test-time temperature scaling into their robustness evaluations to ensure that any observed gains are genuine. Finally, we show how the temperature can be scaled during *training* to improve genuine robustness.

1 Introduction

Deep learning Natural Language Processing (NLP) models are able to perform well in a range of tasks (Manning et al., 2014). However, these NLP models are susceptible to adversarial attacks, where perturbing clean input text samples slightly (accidentally or maliciously by an adversary) can lead to a NLP model misclassifying the perturbed input (Jia and Liang, 2017). However, the emergence of the Adversarial Training (AT) paradigm (Goodfellow et al., 2015) has shown some success in training models to be more robust to these small adversarial perturbations. Here, the traditional training process is adapted to minimize the empirical risk associated with a "robustness loss" as opposed to the risk associated with the standard loss for clean input samples. The robustness loss is the standard loss applied to the worst-case (loss maximizing) adversarial sample for each training sample. In NLP, due to the discrete nature of the text, this adversarial training min-max formulation is particularly challenging as the inner maximization is computationally expensive (Yoo and Qi, 2021). Nevertheless, a variety of approaches have been proposed in the robustness literature, ranging from augmentation of the training set with adversarial examples for a specific model, to sophisticated token-embedding space optimizations for the inner maximization step (Wang et al., 2019b; Goyal et al., 2023).

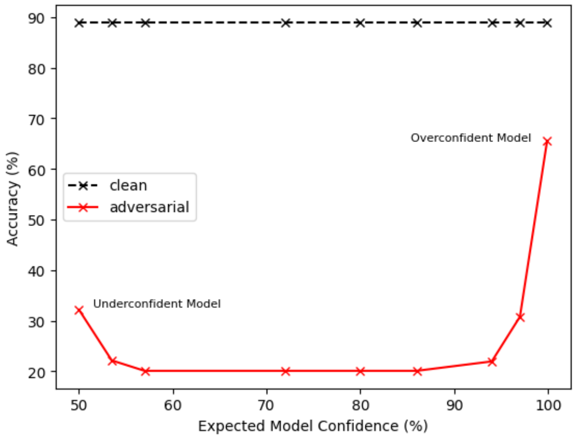


Figure 1: Accuracy on adversarial examples from out-of-the-box adversarial attack for models with different average predicted class confidence, $E_{p(\mathbf{x})}[P_{\hat{\theta}}(\hat{c}|\mathbf{x})]$. Extremely overconfident and underconfident models show increased robustness. We reveal that this increased robustness is merely an *illusion of robustness*.

Although many NLP AT methods appear effective in boosting model robustness, we argue that, in some cases, the observed improvement is merely an *illusion of robustness* (IOR) that can be easily circumvented. In the computer vision literature, Athalye et al. (2018) show that certain modeling decisions can give rise to such an illusion via gra-

[*]Equal contribution. This paper reports on the work VR did whilst at Amazon Web Services. VC holds concurrent appointments as an Amazon Scholar and as a faculty at EPFL. This paper describes the work performed at Amazon.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2500–2525
August 11-16, 2024

dient obfuscation (Papernot et al., 2017). We build on these findings by showing that IORs can also emerge easily in NLP AT due to highly miscalibrated models. Concretely, we:

- Argue that the extreme class confidence of miscalibrated models (Guo et al., 2017) gives rise to an illusion of robustness by disrupting the adversarial attacks' search processes.

- Demonstrate this by intentionally creating highly over- and underconfident models at inference time, and by applying an existing adversarial training technique from the literature. These models appear to be up to three times more robust than the baseline. However, we reveal this robustness to be an illusion that largely evaporates when naive model calibration is applied.

- Further introduce a test-time adversarial temperature optimization algorithm that practically nullifies the perceived robustness gains and demonstrate its efficacy on a range of encoder models, classification datasets, and adversarial attacks.

- Finally use the above insights to improve *true* robustness to unseen attacks by increasing the training temperature, demonstrating its efficacy alone and in combination with other AT methods. In contrast to Papernot et al. (2015), we do not perform model distillation but only training-time temperature scaling as the adversarial defense.

In light of our findings, we urge the community to perform test-time temperature scaling during all robustness evaluations to ensure that the observed robustness is genuine and not merely an illusion. To our knowledge, we are the first to demonstrate this phenomenon in NLP models and propose training-time temperature scaling as an adversarial defense.

2 Background and Related Work

2.1 Adversarial Attacks

An untargeted adversarial attack is able to fool a classification system, $\mathcal{F}()$ with trained parameters $\hat{\theta}$, by perturbing an input sample, $\mathbf{x}$ to generate an adversarial example $\tilde{\mathbf{x}}$ to cause a change in the predicted class,

$$\mathcal{F}(\mathbf{x};\hat{\theta}) \neq \mathcal{F}(\tilde{\mathbf{x}};\hat{\theta}). \tag{1}$$

Traditional adversarial attack definitions (Szegedy et al., 2014) require the perturbation to be *imperceptible* as per human perception. In NLP it can be challenging to measure imperceptibility. Following notation in Raina and Gales (2023) the distance between the original, clean sample and the adversarial example is limited as per a proxy distance measure $\mathcal{G}(\mathbf{x},\tilde{\mathbf{x}}) \leq \epsilon$.

A plethora of adversarial attack approaches have been proposed for efficiently discovering adversarial examples for NLP models (Alzantot et al., 2018; Li et al., 2018; Gao et al., 2018; Wang et al., 2019c; Ren et al., 2019; Jin et al., 2019; Tan et al., 2020; Garg and Ramakrishnan, 2020; Li et al., 2020; Tan and Joty, 2021). Many of the popular attack approaches are implemented in the TextAttack library (Morris et al., 2020).

2.2 Adversarial Training

Standard supervised training methods seek to find model parameters, $\hat{\theta}$, that minimizes the empirical risk (for a dataset of $\mathbf{x} \sim p(\mathbf{x})$), characterised by a loss function,

$$\hat{\theta} = \arg\min_{\theta} \mathop{\mathbb{E}}_{\mathbf{x}\sim p(\mathbf{x})}[\mathcal{L}(\mathbf{x},\theta)]. \tag{2}$$

Adversarial Training (AT; Goodfellow et al., 2015) adapts the objective to minimize the empirical risk associated with the *worst-case* adversarial example, $\tilde{\mathbf{x}}$, such that we are minimizing a *robust loss*,

$$\hat{\theta} = \arg\min_{\theta} \mathop{\mathbb{E}}_{\mathbf{x}\sim p(\mathbf{x})}\left[\max_{\substack{\tilde{\mathbf{x}}:\\ \mathcal{G}(\mathbf{x},\tilde{\mathbf{x}},)\leq\epsilon,\ \tilde{\mathbf{x}}\in\mathcal{A}}} \mathcal{L}(\tilde{\mathbf{x}},\theta)\right]. \tag{3}$$

It is too computationally expensive to perform the inner maximization step to find textual adversarial examples in each step of training. A group of AT methods speed-up this optimization step by finding adversarial examples in the token embedding space, which allows for faster gradient-based approaches: PGD-K (Madry et al., 2018), FreeLB (Zhu et al., 2020), TA-VAT (Li and Qiu, 2020), InfoBERT (Wang et al., 2020). However, limited success of these approaches has been attributed to perturbations in the embedding space being unrepresentative of real textual adversarial attacks. Hence, AT methods such as Adversarial Sparse Convex Combination (ASCC; Dong et al., 2021) and Dirichlet Neighborhood Ensemble (DNE; Zhou et al., 2020) identify a more sensible embedding perturbation space, which they define as the convex hull of word synonyms. Nevertheless, the simplest and most common AT approach in NLP is to augment the training set with textual adversarial examples $\tilde{\mathbf{x}}$ for each clean sample $\mathbf{x}$ by applying the attack to a model trained in the standard manner (Equation 2).

2.3 Model Calibration

Modern deep learning models are often miscalibrated, where the model's confidence in the predicted class does not reflect the ground truth correctness likelihood (Guo et al., 2017). A model with a predicted class confidence $P_{\hat{\theta}}(\hat{c}|\mathbf{x})$, is defined as perfectly calibrated when

$$P\left(\hat{c} = c^{*} | P_{\hat{\theta}}(\hat{c}|\mathbf{x}) = p\right) = p, \quad \forall p \in [0,1], \quad (4)$$

where $\hat{c} = \mathcal{F}(\mathbf{x}; \hat{\theta})$ is the predicted class and the true (label) class is c^{*}. Any deviation from this indicates miscalibration. Typical single-value summaries for the calibration error are the Expected Calibration Error (ECE) and the Maximum Calibration Error (MCE) (Naeini et al., 2015).

2.4 Obfuscated Gradients

Computer vision literature has demonstrated that a 'false sense of security' to adversarial attacks on image classification systems can arise due to 'obfuscated gradients' (Athalye et al., 2018; Papernot et al., 2017; Tramèr et al., 2018). Obfuscated gradients block an adversary's search process for an adversarial example and hence give the sense that the system is robust to the adversarial attack — however, this is not the case, as adversarial examples still exist, but specific gradient-based mechanisms used by adversaries to find the adversarial examples are not effective when obfuscated gradients are present. In computer vision, obfuscated gradients in typical image systems can arise due to various reasons, including 'shattered gradients', 'stochastic gradients', exploding/vanishing gradients and any form of gradient masking (Athalye et al., 2018; Papernot et al., 2017; Tramèr et al., 2018). However, to our knowledge, no previous work has explored how the illusion of robustness phenomenon emerges in NLP systems.

In this work, we observe that in various adversarial training regimes designed for NLP tasks, there is often the risk that systems become extremely miscalibrated. Extreme miscalibration effectively results in 'obfuscated gradients', which in turn results in the Illusion of Robustness. Hence, in summary, both computer vision and NLP systems can suffer from the Illusion of Robustness due to obfuscated gradients, but the practical scenarios which cause these obfuscated gradients differ for image and NLP classification systems. As a result, the methods used to mitigate the root causes behind obfuscated gradients for image classification and NLP classification systems, also differ.

3 The Illusion of Robustness

Certain modeling approaches can lead to an illusion of robustness (IOR). In computer vision, it is shown that obfuscating gradients (Athalye et al., 2018) through *shattered*, *stochastic* or *exploding/-vanishing* gradients; or masking gradients (Papernot et al., 2017; Tramèr et al., 2018) can lead to such an IOR - the model appears robust to adversarial attacks. We build on these works and argue that (un)intentional extreme model miscalibration is an example of a realistic cause of IOR.

The robustness gains observed for traditional NLP AT approaches (Equation 3), may not always be due to inherent robustness gains, but can be a consequence of extreme model miscalibration. This miscalibration can induce extreme confidence predictions, such that the model's predicted class confidence $P_{\hat{\theta}}(\hat{c}|\mathbf{x})$ is either very high (overconfident) or very low (underconfident). Figure 1 (using a standard NLP model, test dataset and adversarial attack described in Section 3.3) demonstrates that highly miscalibrated models with extreme confidence values in the predicted class (around 1.0 for overconfident models or $1/C$, where C is the number of classes for underconfident models) are significantly more "robust" to adversarial attacks.

This apparent increase in robustness of extremely miscalibrated models can be explained. For both underconfident and overconfident models, the predicted class confidence has very little variance for different input sequences, $\mathbf{x}$,

$$E_{p(\mathbf{x})}[P_{\hat{\theta}}(\hat{c}|\mathbf{x}) - \mathbb{E}_{p(\mathbf{x})}[P_{\hat{\theta}}(\hat{c}|\mathbf{x})]]^{2} < \zeta, \quad (5)$$

where ζ is some small variance. The narrow confidence distribution makes it challenging for an adversary to identify an appropriate search direction for adversarial examples. To illustrate this, consider a miscalibrated model with extremely high confidence in the predicted class probability, $P_{\hat{\theta}}(\hat{c}|\mathbf{x}) \approx 1.0$, then for most search directions $\mathbf{d}$ that are not in an adversarial direction $\mathbf{d} \neq \tilde{\mathbf{d}}$ (where $\tilde{\mathbf{x}} = \mathbf{x} + \tilde{\mathbf{d}}$) the model has very little sensitivity,[1] i.e.,

$$\mathbf{d}^{T} \nabla_{\mathbf{x}} P_{\hat{\theta}}(\hat{c}|\mathbf{x}) \approx 0. \quad (6)$$

[1] Note that these strict mathematical operations are not

Consequently, any whitebox adversarial attack approach looking to exploit gradients or even a blackbox attack approach measuring the sensitivity of the predicted probability, has a small confidence range to observe. The implication is that the impact of any proposed perturbation gives a very *noisy* signal to its actual effect on the output. As a result, the adversarial attack search process will converge extremely slowly or outright fail to find the desired adversarial perturbation direction $\tilde{\mathbf{d}}$. We verify this hypothesis empirically in Appendix E.7, corroborating related computer vision literature on gradient obfuscation (Athalye et al., 2018). We first demonstrate how to induce an IOR by explicitly miscalibrating models at test-time, before discussing how this can happen unintentionally during training.

3.1 Explicit: Test-time Temperature Scaling

Let $\hat{\theta}$ be a model trained using the standard training objective, as in Equation 2. For this model with predicted logits, $l_1, \ldots, l_C$ for C output classes, the probability of a specific class is typically estimated by the Softmax function, $P_{\hat{\theta}}(c|\mathbf{x}) = \frac{\exp{(l_c)}}{\sum_i \exp{(l_i)}}$. However, we can intentionally miscalibrate the model and increase the model confidence at *test time* by using a temperature, $T = T_d$, to scale the predicted logits,

$$P_{\hat{\theta}}(c|\mathbf{x}; T) = \frac{\exp{(l_c/T)}}{\sum_i \exp{(l_i/T)}}. \tag{7}$$

A design choice of $T_d \ll 1.0$ concentrates the probability mass in the largest logit class to create an *overconfident* model, whilst conversely $T_d \gg 1.0$ creates an *underconfident* model. Hence, explicitly setting a design temperature $T^{(d)}$ at inference time can be used to serve highly miscalibrated models, which can disrupt an adversary's attack search process (Equation 6), whilst maintaining the simplicity of the standard training objective (Equation 2).

3.2 Implicit Overconfidence: Grad. Norm.

Having shown how to induce an IOR by explicitly scaling a model's temperature, we now discuss how certain implementation strategies and algorithmic features in adversarial training (AT) procedures can also implicitly induce model overconfidence.

We first examine the recently proposed Danskin Descent Direction approach for adversarial training (DDi-AT; Latorre et al., 2023). Latorre et al. (2023) adapted the standard AT paradigm (Equation 3) to identify optimal gradient update directions for increased robustness, showing promising results in computer vision. We describe the NLP-specific implementation in Appendix A.

In preliminary experiments, we observed that DDi-AT creates highly overconfident models without compromising clean accuracy, such that a DDi-AT model almost always predicts with near 100% confidence in its predicted class, $P_{\hat{\theta}}(c|\mathbf{x}) \approx 1.0$ (Table 1). Further ablations, detailed in Appendix A.3, revealed that the gradient normalization step in the DDi algorithm was responsible for this model overconfidence. In the following section, we verify this by applying gradient normalization to other AT schemes that may use it during training.

3.3 Experiments

We first present the experimental setup for all experiments in this paper before showing how illusions of robustness can arise when evaluating highly overconfident or under-confident models for robustness.

Data. Experiments are carried out on six standard NLP classification datasets. For IOR experiments we use Rotten Tomatoes (Pang and Lee, 2005); the Twitter Emotions Dataset (Saravia et al., 2018); and the AGNews dataset (Zhang et al., 2015). We observe the same general trends across all datasets, and therefore present the results on Rotten Tomatoes here and include the others in Appendix E.1.

Models. We follow existing adversarial robustness literature and use Transformer (Vaswani et al., 2017) encoders, which are state-of-the-art on many classification tasks. [2] Specifically, we consider the base variants of DeBERTa (He et al., 2020), RoBERTa (Liu et al., 2019) and BERT (Devlin et al., 2019). We observe the same general trends across all models, and therefore present the results for DeBERTa here and the others in Appendix E.2.

Adversarial attacks. We experiment with four common adversarial attacks. BERT-based Adversarial Examples (*bae*) (Garg and Ramakrishnan, 2020) is a word-level blackbox attack, where the adversary only has access to the model inputs and predictions. We also include Textfooler (*tf*) (Jin et al., 2019) and Probability Weighted Word Saliency

defined for the input text space and are simply representative of equivalent discrete textual space perturbations.

[2] Appendix C demonstrates the superior performance of encoder-only models relative to generative language models for many classification tasks.

Intv. Type	Method	clean	$\bar{P}(\hat{c}\vert\mathbf{x}_{\textbf{clean}})$	$\bar{P}(\hat{c}\vert\mathbf{x}_{\textbf{adv}})$
None	baseline	88.96 ±0.30	97.08 ±0.26	86.04 ±0.68
	pgd	88.24 ±0.73	97.56 ±0.42	87.25 ±0.59
	ascc	87.77 ±0.36	97.50 ±0.24	86.99 ±0.43
Explicit	↓conf	88.96 ±0.30	50.00007 ±0.00	50.00004 ±0.00
	↑conf	88.96 ±0.30	99.98 ±0.02	99.95 ±0.01
Implicit	baseline*	88.56 ±0.19	99.96 ±0.04	99.93 ±0.02
	ddi-at	87.90 ±0.49	99.97 ±0.03	99.91 ±0.01
	pgd*	88.59 ±0.64	99.96 ±0.04	99.90 ±0.01
	ascc*	87.77 ±0.36	99.97 ±0.04	99.92 ±0.01

Table 1: Clean accuracy (%) and model confidence (%) on clean and adv. examples from Rotten Tomatoes for extreme confidence systems. We categorize them by intervention type — explicit temperature scaling and implicit induction via gradient normalization during training. We use pwws to generate the adv. examples.

Method	clean	bae	tf	pwws	dg
baseline	88.96 ±0.30	31.39 ±1.20	17.82 ±0.49	20.42 ±0.62	20.11 ±0.94
dg-aug	87.12 ±0.39	34.74 ±1.59	22.36 ±1.83	26.11 ±2.57	37.43 ±0.75
pgd	88.24 ±0.73	33.65 ±0.57	19.92 ±0.47	26.70 ±0.87	26.05 ±0.61
ascc	87.77 ±0.36	33.61 ±0.64	15.13 ±2.17	23.50 ±0.77	26.80 ±2.11
freelb	88.74 ±0.32	32.52 ±0.52	19.51 ±1.70	24.55 ±0.70	24.52 ±0.73
↓conf (§3.1)	88.96 ±0.30	31.21 ±0.94	20.98 ±0.99	25.17 ±0.89	32.18 ±2.78
↑conf (§3.1)	88.96 ±0.30	37.71 ±1.18	54.35 ±0.73	59.29 ±0.62	65.60 ±1.81
baseline*	88.56 ±0.19	33.71 ±3.57	47.22 ±5.01	53.03 ±3.03	59.10 ±4.09
ddi-at (§3.2)	87.90 ±0.49	39.18 ±0.75	56.54 ±1.67	61.07 ±0.99	66.73 ±1.01
pgd*	88.59 ±0.64	39.94 ±0.55	58.02 ±1.04	64.45 ±0.77	67.02 ±0.83
ascc*	87.77 ±0.36	40.01 ±0.69	54.32 ±1.57	63.99 ±0.86	67.43 ±0.93

Table 2: Accuracy (%) of extreme confidence systems on Rotten Tomatoes compared to standard AT methods under various adversarial attacks.

(*pwws*) (Ren et al., 2019), more powerful word-level attacks. Finally, we include the DeepWordBug (*dg*) (Gao et al., 2018) attack as a whitebox, *character*-level adversarial attack. We use the TextAttack implementations and their default settings (Morris et al., 2020). To measure the impact of each adversarial attack, we report the *adversarial accuracy*, which is the accuracy of the target model on adversarial examples found by the attack.

Explicit temperature scaling. We create under- and over-confident models, ↓*conf* and ↑*conf*, by scaling the temperature as in Section 3.1.

AT approaches. We consider a range of AT methods: Danskin Descent Direction (*ddi-at*; Latorre et al., 2023), PGD-K (*pgd*; Madry et al., 2018) and FreeLB (*freelb*; Zhu et al., 2020 as embedding-space AT schemes, and ASCC (*ascc*; Dong et al., 2021) as a text-embedding combined AT approach. We further create variants of the baseline, *pgd* and *ascc* that use gradient normalization during training, named *baseline**, *pgd**, and *ascc**, respectively. Finally, we consider the most common NLP AT method: augmenting the training set with adversarial examples, using DeepWordBug as the representative attack (*dg-aug*). Hyperparameters are detailed in Appendix D.

3.3.1 Results

Model Confidence. We see from Table 1 that the ↑*conf*, *baseline**, *ddi-at*, *pgd**, and *ascc** models are significantly more confident than the models with no intervention, whilst the ↓*conf* model is far less confident, as intended. Note that the differences in the confidence are more pronounced for the adversarial examples (*pwws* is used to attack the test set). The differences in confidence between the original and gradient normalization variants of the *baseline*, *pgd*, and *ascc* models verifies our hypothesis that using gradient normalization during training is one cause of extreme overconfidence.

Robustness. Table 2 presents the adversarial robustness of each model as measured by the adversarial accuracy under the different adversarial attacks. While the regular AT approaches (*dg-aug*, *pgd*, *ascc*, *freelb*) increase robustness to some extent, the gains in robustness of the highly overconfident models (↑*conf*, *ddi-at*, *baseline**, *pgd**, *ascc**) **appear** to outstrip them by two- to three-fold.

We argue that the apparent increase in robustness of the extreme confidence models (↓*conf*, ↑*conf*, *baseline**, *ddi-at*, *pgd** and *ascc**) in Table 2 is due to the attack search process being disrupted, but the models are still susceptible to adversarial examples. We know this must be true for the explicitly temperature-scaled models since the predicted class never changes from the *baseline*'s, only its confidence. We further empirically find that extreme confidence results in a noisier search for the adversarial attack. We discuss this in greater detail in Appendix E.7 due to space limitations.

4 Piercing the Illusion

Having demonstrated how to induce an IOR from the developer or defender's perspective, we now discuss how to nullify it as an adversary. Although the following approaches involve modifying aspects of the model's output at test-time, these modifications are only used to create/find adversarial examples, which can then be applied to the original, un-modified model served by the model developer.

4.1 Naive Test-Time Temperature Calibration

Highly miscalibrated models (Section 3) interfere with an adversarial attack's ability to find meaningful search directions due to the little sensitivity in the predicted probabilities. An adversary aims to mitigate this disruption to the attack search process. The natural solution, then, is to calibrate the model so that the confidences are in a sensible range and can now be exploited by adversarial attacks.

A strong indicator of model miscalibration (Section 2.3) can be given by the Negative Log Likelihood (NLL; Hastie et al., 2017). Thus, assuming access to the output model logits $l_1, \ldots, l_C$ and a labelled validation set of data $\{\mathbf{x}_i, c_i^*\}_i$, an adversary can apply test-time temperature calibration (Guo et al., 2017). This works regardless of *how* the model was miscalibrated (implicitly or explicitly). Now, the adversary optimizes an adversarial temperature, T_a to minimize the Negative Log Likelihood (NLL) of the validation set samples,

$$T_a = \arg\min_{T} \sum_i -\log P_{\hat{\theta}}(c_i^*|\mathbf{x}_i; T), \tag{8}$$

where $P_{\hat{\theta}}(c^*|\mathbf{x}; T)$ is the confidence of the true class after temperature scaling as in Equation 7. Due to the continuous nature of the transformation and the need to optimize a single parameter, T_a, we use standard gradient descent optimization.[3]

Other than temperature optimization, an adversary can attempt other post-training model calibration approaches such as Histogram Binning (Zadrozny and Elkan, 2001), isotonic regression (Zadrozny and Elkan, 2002) and multi-class versions of Platt scaling (Niculescu-Mizil and Caruana, 2005; Platt and Karampatziakis, 2007). However, we empirically find temperature calibration to be the most practical and effective for an adversary seeking to mitigate a model's IOR. We discuss this in greater detail in Appendix E.6.

4.2 Adversarial Temperature Optimization

While simple, the naive calibration approach has two shortcomings:

1. The adversarial temperature, T_a is not directly tuned to minimize adversarial robustness, as it only considers the likelihood of clean examples in a validation set.

2. Learning the adversarial temperature, T_a to minimize the NLL (Equation 8) uses a gradient descent–based optimization algorithm that is sensitive to hyperparameters and does not guarantee an optimal solution.

Hence, we now outline an algorithm that directly optimizes the adversarial temperature T_a to minimize a model's adversarial robustness at test time. We define the adversarial accuracy, $\mathcal{Q}()$ as a function of the temperature parameter,

$$\mathcal{Q}(T) = \frac{1}{J} \sum_j \mathbb{I}\left[\mathcal{F}(\tilde{\mathbf{x}}_j(T)) = c_j^*\right], \tag{9}$$

where $\tilde{\mathbf{x}}_j(T)$ represents the adversarial example generated from an adversarial attack on the given model, $\hat{\theta}$ with the logits scaled by a temperature T as in Equation 7. Figure 1 illustrates that as the temperature parameter is swept from large to small values (increasing model confidence), the adversarial accuracy, $\mathcal{Q}()$ behaves almost as a convex function of temperature, T, such that, $\mathcal{Q}(\alpha T_1 + (1-\alpha)T_2) \leq \alpha\mathcal{Q}(T_1) + (1-\alpha)\mathcal{Q}(T_2)$, where $0 \leq \alpha \leq 1$. The optimal adversarial temperature T_a minimizes the adversarial accuracy $\mathcal{Q}(T)$,

$$T_a = \arg\min_{T} \mathcal{Q}(T). \tag{10}$$

Hence, T_a can be found efficiently over the non-differentiable convex function, $\mathcal{Q}()$, using search methods such as golden section search (Kiefer, 1953). We use the Brent-Dekker method, an extension of golden section search that accounts for a parabolic convergence point (Brent, 1971).

4.3 Experiments

Setup. We maintain the experimental setup as Section 3.3 and supplementary results for different models and datasets are provided in Appendix E.

IOR mitigation. We experiment with the two proposed approaches to mitigate the disruption of the adversarial attack search processes and remove

[3] Inspired by https://github.com/gpleiss/temperature_scaling/tree/master.

Method	Adv.	clean	bae	tf	pwws	dg
baseline	-	88.96 ±0.30	31.39 ±1.20	17.82 ±0.49	20.42 ±0.62	20.11 ±0.94
↓conf	-	88.96 ±0.30	31.21 ±0.94	20.98 ±0.99	25.17 ±0.89	32.18 ±2.78
	cal	88.96 ±0.30	31.52 ±0.34	21.89 ±0.43	27.58 ±1.31	31.52 ±0.34
	opt	88.96 ±0.30	31.44 ±1.15	17.82 ±0.49	20.86 ±0.64	21.98 ±1.66
↑conf	-	88.96 ±0.30	37.71 ±1.18	54.35 ±0.73	59.29 ±0.62	65.60 ±1.81
	cal	88.96 ±0.30	31.39 ±1.20	17.82 ±0.49	20.45 ±0.74	21.64 ±1.46
	opt	88.96 ±0.30	31.39 ±1.20	17.82 ±0.49	20.90 ±0.94	21.06 ±0.82
baseline*		88.56 ±0.19	33.71 ±3.57	47.22 ±5.01	53.03 ±3.03	59.10 ±4.09
	cal	88.56 ±0.19	32.40 ±0.14	18.79 ±0.48	21.36 ±1.22	21.11 ±0.66
ddi-at	-	87.90 ±0.49	39.18 ±0.75	56.54 ±1.67	61.07 ±0.99	66.73 ±1.01
	cal	87.90 ±0.49	31.80 ±0.57	18.36 ±3.01	23.08 ±1.96	22.89 ±3.38
	opt	87.90 ±0.49	31.80 ±0.57	18.88 ±3.32	22.16 ±1.03	22.28 ±1.12
pgd*	-	88.59 ±0.64	39.94 ±0.55	58.02 ±1.04	64.45 ±0.77	67.02 ±0.83
	cal	$\underline{88.59}$ ±0.64	33.71 ±0.20	17.73 ±0.86	25.20 ±1.80	25.74 ±1.46
ascc*	-	87.77 ±0.36	40.01 ±0.69	54.32 ±1.57	63.99 ±0.86	67.43 ±0.93
	cal	87.77 ±0.36	33.61 ±0.64	15.13 ±2.17	23.50 ±0.77	26.80 ±2.11

Table 3: Clean and adv. accuracy (%) on Rotten Tomatoes after mitigating the *illusion of robustness* of highly miscalibrated systems using temperature calibration (*cal*) or adversarial temperature optimization (*opt*).

the IOR: naive temperature calibration (*cal*) and adversarial temperature optimization (*opt*). The learning rate is set to 0.01 with a maximum of 5000 iterations for *cal*. For *opt*, we use DeepWordBug to attack the validation set when optimizing for T_a.

Evaluation. We first modify the model by scaling the predicted logits by T_a and then the adversarial attacks are run on the modified model to find adversarial examples. The original, unmodified model is lastly evaluated on these adversarial examples.

4.3.1 Results

Table 3 shows the impact of the different adversarial approaches (*cal* and *opt*) to learn T_a on the adversarial robustness of the models. For the overconfident models, ↑*conf*, *baseline**, *ddi-at*, *pgd** and *ascc**, simple temperature calibration (*cal*) is sufficient to cause a significant drop in model robustness.[4] For the ↓*conf* model, the temperature optimization approach (*opt*) is necessary to significantly reduce robustness, illustrating its efficacy at nullifying IORs.

It is worth analyzing why simple calibration (*cal*) is effective in removing IOR in the overconfident models (↑conf) but not for the underconfident models (↓conf). This observation can perhaps be explained by considering the search space for the calibrating temperature: the *cal* method is naive temperature calibration, where a temperature parameter T_a (divisor of logits) is learnt on a validation set by minimizing the negative log likelihood. The solution for T_a differs when calibrating low-confidence and high-confidence models. For low-confidence models, T_a must lie between 0 and 1 (as the logits have to be scaled up), and for high-confidence models, T_a only needs to be greater than 1. Hence, the solution space for T_a is more constrained when calibrating low-confidence models, meaning it is more challenging for gradient based search methods to find the solution space for T_a. Therefore, *cal* struggles more when calibrating the low confidence models.

Critically, our results highlight the risk of some AT approaches, whether by design or as an implementation detail, giving the illusion of robustness even if they do not yield genuinely robust models.

4.4 Discussion

Although adversarial temperature optimization is more effective in nullifying the IOR, it is significantly slower than the naive calibration approach. Therefore, naive temperature calibration is favorable in computational resource–scarce settings and should always be run at minimum in robustness evaluations. Another consideration is detectability. When evaluating an API model, adversarial temperature optimization will require sending many similar queries as part of the attack, which has a higher chance of being detected and blocked by the API. In contrast, naive temperature calibration only requires querying the API for predictions on clean examples, which will appear far more innocuous.

One might even argue that to expose an IOR, it is unnecessary for an adversary to modify the model with adversarial temperature scaling to find adversarial examples. Instead, we could find adversarial examples for another model (e.g., *baseline*) and transfer to the target model. We test this hypothesis in Appendix E.3. We find that although the transfer attack from *baseline* to *ddi-at* reduces the adversarial accuracy, it is unable to bring it down to the same values as *baseline*, as achieved by our proposed temperature scaling approaches.

[4] Appendix E.4 discusses the relationship between the calibration error and the model confidence.

5 Raising the Training Temperature for Genuine Robustness

Section 3 showed that it is easy to unintentionally develop AT schemes that do not yield true robustness gains but instead induce IOR. The success of temperature scaling at interfering with an attack's search process poses the natural question: Can a similar effect be induced in training such that it cannot be nullified, thus improving true robustness?

We now present a simple modification to standard training that boosts the true robustness of NLP models to unseen attacks. We consider an attack to be unseen when its adversarial examples are not used in adversarial training. In our experiments, this refers to all AT methods other than *dg-aug*.

5.1 Method

Since scaling the temperature down at test time reduces an adversarial attack's efficacy, intuitively, we would like to "bake" this behavior into a model's weights such that it cannot be neutralized at the logit layer by the approaches from Section 4.

We propose to do this by increasing the temperature during training, bringing the probabilities of the different classes closer together. This encourages the model's parameters to compensate by pushing the logits of the different classes further apart (Figure 3 in the Appendix). This increases the distance to the class boundary in the logit space and makes it more difficult for an adversarial attack to change the predicted class (Robey et al., 2023).

Adversarial robustness can also be viewed as type of generalization, and Agarwala et al. (2020) find that model generalization depends strongly on the training temperature, where larger temperatures yield stronger results for vision models. Therefore, our method can also be viewed as flattening the loss landscape, which has been shown to improve generalization for adversarial robustness in computer vision (Stutz et al., 2021). Future work will aim to rigorously understand the observed empirical robustness gains of high temperature training.

Method	Clean	bae	tf	pwws	dg
baseline*	88.56 ±0.19	32.40 ±0.14	18.79 ±0.48	21.36 ±1.22	21.11 ±0.66
$\oplus T$	87.55 ±0.44	35.83 ±0.84	26.83 ±4.57	31.49 ±3.07	35.18 ±4.71
pgd*	88.59 ±0.64	33.71 ±0.20	17.73 ±0.86	25.20 ±1.80	25.74 ±1.46
$\oplus T$	87.77 ±0.43	34.77 ±0.33	24.55 ±1.76	31.46 ±1.08	31.77 ±2.64
freelb*	88.74 ±0.32	32.52 ±0.52	19.51 ±1.70	24.55 ±0.70	24.52 ±0.73
$\oplus T$	88.02 ±0.52	35.15 ±0.80	25.17 ±0.96	29.96 ±0.68	31.49 ±1.04
ascc*	87.77 ±0.36	33.61 ±0.64	15.13 ±2.17	23.50 ±0.77	26.80 ±2.11
$\oplus T$	86.36 ±0.80	34.93 ±1.12	27.36 ±0.72	30.93 ±1.38	33.46 ±1.65
dg-aug*	87.12 ±0.39	34.74 ±1.59	22.36 ±1.83	26.11 ±2.57	37.43 ±0.75
$\oplus T$	87.09 ±0.22	36.99 ±2.64	26.92 ±2.86	31.43 ±1.67	36.40 ±1.90

Table 4: Adversarial Training (AT) combined with a training temperature of $T = 200$ ($\oplus T$) on Rotten Tomatoes. We report the post-calibration (*cal*) accuracy. The higher accuracy between the AT model and the AT$\oplus T$ model is underlined. The higher training temperature always improves robustness to unseen adversarial attacks.

5.2 Experiments

We maintain the setup from Section 3.3, with the addition of three datasets from the General Language Understanding Evaluation benchmark (GLUE; Wang et al., 2019a): the Corpus of Linguistic Acceptability; Question-answering NLI; and the Microsoft Research Paraphrase Corpus. We further include gradient normalization (*) in all experiments, as we found this to stave off decreases in clean accuracy as we increase training temperature (ablation is detailed in Appendix F.7). To ensure the robustness gains are not IORs, we apply the test-time calibration described in Section 4. Appendix F.8 demonstrates the extent of IOR when such test-time temperature scaling is not applied.

5.2.1 Results

Results in Table 4 show that a high training temperature not only boosts genuine adversarial robustness for a model trained with the standard objective (*baseline*$^*\oplus T$), but can also be combined with popular adversarial AT schemes for a further boost. We observe that the high training temperature approach improves the adversarial accuracy against unseen attacks for all of the adversarial training approaches experimented upon ($\oplus T$). This demonstrates that high temperature training is complementary to existing AT methods and consistently encourages a gain in genuine robustness. Interestingly, we observe that for *dg-aug**, using a high temperature during adversarial training further improves adversarial accuracy for the unseen attacks but causes a slight drop in adversarial accuracy for the (seen) *dg* attack compared to regular AT. This suggests that although a high training temperature successfully increases robustness to unseen attacks, it may not yield further robustness gains against seen attacks.

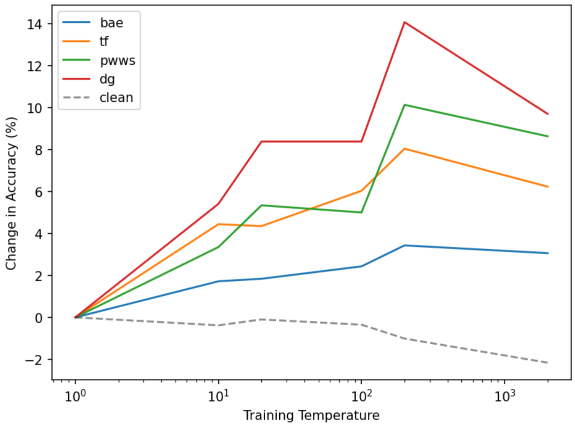


Figure 2: Change in post-calibration accuracy on Rotten Tomatoes as training temperature varies. We observe that a higher temperature during training increases robustness against unseen attacks (*bae, tf, pwws, dg* here). The change in adversarial accuracy relative to the baseline ($T = 1$) demonstrates the increase in robustness.

Additionally, Figure 2 presents the change in clean and adversarial accuracy of a model trained with the standard training objective and different temperatures T used during training.[5] We further observe a consistent robustness profile where robustness peaks at similar temperatures for all tested attack types. This is particularly useful for a model developer with access to only one type of adversarial attack. The training temperature can be tuned for optimal robustness on that specific attack form with the confidence that the robustness gains will transfer to the other unseen/unknown attack forms. Finally, we observe a decrease in accuracy at extremely large training temperatures, an indication that overly large temperatures may excessively smooth the predicted probability distribution and make it too challenging for the model to learn. However, this is easy to avoid by sweeping the temperature with a single attack since there is a consistent robustness profile for each dataset.

6 Conclusion

NLP models are susceptible to adversarial attacks, where small changes in the input cause the model to predict incorrectly. Many adversarial training (AT) approaches have been proposed to induce robustness to adversarial attacks. In this work, we argued that the observed robustness gains may not be due to true increases in model robustness. We demonstrated how AT schemes can unknowingly create highly miscalibrated models that disrupt common adversarial attack search methods by obfuscating gradients, yielding up to three-fold perceived robustness gains. However, this is merely an *illusion of robustness* (IOR). We proposed simple methods an adversary could use to circumvent such gains. Specifically, we showed how to perform test-time temperature scaling to mitigate disruptions to the adversarial attack search processes and pierce the IOR. Hence, we strongly recommend all adversarial robustness evaluations incorporate adversarial temperature scaling to ensure any observed robustness gain is genuine and not an *illusion*. Finally, we proposed a practical training-time modification to increase a model's genuine robustness to unseen adversarial attacks and demonstrated its efficacy alone and in combination with other AT methods.

7 Limitations

- Empirical results are presented for state-of-the-art encoder-based Transformer models. Although Appendix C demonstrates that encoder-based models are more appropriate for many NLP classification tasks than the recently popularized generative Large Language Models (LLMs), it would be useful to investigate how susceptible these larger, decoder-based LLMs are to the IOR.

- We evaluate common Adversarial Training (AT) baselines to illustrate the IOR phenomenon. However, future work would benefit from conducting an in-depth study that includes other recently proposed approaches for adversarial robustness, e.g., contrastive learning based approaches (Rim et al., 2021) and Textual Manifold Defence (Nguyen Minh and Luu, 2022), where all inputs are mapped to a robust manifold. This would help the community understand the extent to which these proposed approaches are improving true robustness and the extent to which they may be unknowingly creating an IOR.

- In Section 5, we used a constant temperature during training for simplicity. However, varying the temperature over the course of training may further increase the effectiveness of high temperature training. Future work will aim to study the impact of temperature schedules.

[5] Appendix F.5 contains a detailed breakdown for each training temperature, adversarial attack, and dataset.

8 Risks and Ethics

This work presents results on the topic of adversarial training. The contributions in this work encourage the development of truly robust systems and therefore there are no identified ethical concerns.

References

Atish Agarwala, Jeffrey Pennington, Yann N. Dauphin, and Samuel S. Schoenholz. 2020. Temperature check: theory and practice for training models with softmax-cross-entropy losses. *CoRR*, abs/2010.07344.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. pages 2890–2896.

Anish Athalye, Nicholas Carlini, and David A. Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *CoRR*, abs/1802.00420.

R. P. Brent. 1971. An algorithm with guaranteed convergence for finding a zero of a function. *The Computer Journal*, 14(4):422–425.

Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. 2018. On the intriguing connections of regularization, input gradients and transferability of evasion and poisoning attacks. *CoRR*, abs/1809.02861.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. pages 4171–4186.

Xinshuai Dong, Anh Tuan Luu, Rongrong Ji, and Hong Liu. 2021. Towards robustness against natural language word substitutions. *CoRR*, abs/2107.13541.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. *CoRR*, abs/1801.04354.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, Online. Association for Computational Linguistics.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples.

Shreya Goyal, Sumanth Doddapaneni, Mitesh M. Khapra, and Balaraman Ravindran. 2023. A survey of adversarial defences and robustness in nlp.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. pages 1321–1330.

Trevor Hastie, Jerome Friedman, and Robert Tisbshirani. 2017. *The elements of Statistical Learning: Data Mining, Inference, and prediction*. Springer.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced BERT with disentangled attention. *CoRR*, abs/2006.03654.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is BERT really robust? natural language attack on text classification and entailment. *CoRR*, abs/1907.11932.

J. Kiefer. 1953. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, 4(3):502–506.

Fabian Latorre, Igor Krawczuk, Leello Tadesse Dadi, Thomas Pethick, and Volkan Cevher. 2023. Finding actual descent directions for adversarial training. In *The Eleventh International Conference on Learning Representations*.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. *CoRR*, abs/1812.05271.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial attack against BERT using BERT. pages 6193–6202.

Linyang Li and Xipeng Qiu. 2020. Textat: Adversarial training for natural language understanding with token-level perturbation. *CoRR*, abs/2004.14543.

Zongyi Li, Jianhan Xu, Jiehang Zeng, Linyang Li, Xiaoqing Zheng, Qi Zhang, Kai-Wei Chang, and Cho-Jui Hsieh. 2021a. Searching for an effective defender: Benchmarking defense against adversarial word substitution.

Zongyi Li, Jianhan Xu, Jiehang Zeng, Linyang Li, Xiaoqing Zheng, Qi Zhang, Kai-Wei Chang, and Cho-Jui Hsieh. 2021b. Searching for an effective defender: Benchmarking defense against adversarial word substitution. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3137–3147, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2019. Towards deep learning models resistant to adversarial attacks.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. *Proceedings of the ... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, 2015:2901–2907.

Dang Nguyen Minh and Anh Tuan Luu. 2022. Textual manifold-based defense against natural language adversarial examples. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6612–6625, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 625–632, New York, NY, USA. Association for Computing Machinery.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, page 506–519, New York, NY, USA. Association for Computing Machinery.

Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2015. Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR*, abs/1511.04508.

Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063.

Francesco Periti, Haim Dubossarsky, and Nina Tahmasebi. 2024. (chat)gpt v bert: Dawn of justice for semantic change detection.

John Platt and Nikos Karampatziakis. 2007. Probabilistic outputs for svms and comparisons to regularized likelihood methods.

Vyas Raina and Mark Gales. 2023. Sample attackability in natural language adversarial attacks.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

Daniela N. Rim, DongNyeong Heo, and Heeyoul Choi. 2021. Adversarial training with contrastive learning in nlp.

Alexander Robey, Fabian Latorre, George J. Pappas, Hamed Hassani, and Volkan Cevher. 2023. Adversarial training should be cast as a non-zero-sum game.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium. Association for Computational Linguistics.

David Stutz, Matthias Hein, and Bernt Schiele. 2021. Relating adversarially robust generalization to flat minima. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7807–7817.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks.

Samson Tan and Shafiq Joty. 2021. Code-mixing on sesame street: Dawn of the adversarial polyglots. In *Proceedings of the 2021 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3596–3616, Online. Association for Computational Linguistics.

Samson Tan, Shafiq Joty, Min-Yen Kan, and Richard Socher. 2020. It's morphin' time! Combating linguistic discrimination with inflectional perturbations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2920–2935, Online. Association for Computational Linguistics.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2018. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. Glue: A multi-task benchmark and analysis platform for natural language understanding.

Boxin Wang, Shuohang Wang, Yu Cheng, Zhe Gan, Ruoxi Jia, Bo Li, and Jingjing Liu. 2020. Infobert: Improving robustness of language models from an information theoretic perspective. *CoRR*, abs/2010.02329.

William Yang Wang, Sameer Singh, and Jiwei Li. 2019b. Deep adversarial learning for NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 1–5, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiaosen Wang, Hao Jin, and Kun He. 2019c. Natural language adversarial attacks and defenses in word level. *CoRR*, abs/1909.06723.

Jin Yong Yoo and Yanjun Qi. 2021. Towards improving adversarial training of NLP models. *CoRR*, abs/2109.00544.

Bianca Zadrozny and Charles Elkan. 2001. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 609–616, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 694–699, New York, NY, USA. Association for Computing Machinery.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.

Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. 2023. Can chatgpt understand too? a comparative study on chatgpt and fine-tuned bert.

Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-Wei Chang, and Xuanjing Huang. 2020. Defense against adversarial attacks in NLP via dirichlet neighborhood ensemble. *CoRR*, abs/2006.11627.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. Freelb: Enhanced adversarial training for natural language understanding.

A Danksin's Descent Direction for NLP

A.1 Original Theory

Latorre et al. (2023) demonstrate that the standard formulation and implementation of AT (as in Equation 3) is potentially flawed. Specifically, solving the inner maximization to find the *worst-case* adversarial example $\tilde{\mathbf{x}}$, can give a gradient direction (in standard stochastic gradient descent approaches), that can in fact *increase* the robust loss (the new worst-case adversarial example, $\tilde{\mathbf{x}}$, with the updated model parameters, θ, can give a robust loss that is greater than before the update step), i.e. worsening the adversarial robustness of the model. This flaw is attributed to the reliance on a single adversarial example, as a parameter gradient step to reduce the model's sensitivity to a particular adversarial example does not guarantee reduction in the model's sensitivity to all adversarial examples (the model may now be less robust to other adversarial examples) for a specific sample $\mathbf{x}$. The paper argues that their exist multiple solutions to the inner-maximization for the robust loss and the optimal parameter gradient direction depends on all of those solutions. Thus, Equation 3 can theoretically be adapted to selecting the adversarial example that maximises the gradient direction in each gradient update step for a batch size of K samples,

$$\theta_{i+1} = \Phi\left(\theta_i, \boldsymbol{\gamma}^* = -\frac{\nabla_\theta g(\mathbf{x}_{1:K}, \theta_i, \hat{\tilde{\mathbf{x}}}_{1:K})}{||\nabla_\theta g(\mathbf{x}_{1:K}, \theta_i, \hat{\tilde{\mathbf{x}}}_{1:K})||_2}\right),$$
$$g(\mathbf{x}_{1:K}, \theta_i, \hat{\tilde{\mathbf{x}}}_{1:K}) = \frac{1}{K}\sum_k \mathcal{L}(\hat{\tilde{\mathbf{x}}}_k, \theta_i),$$
$$\hat{\tilde{\mathbf{x}}}_k = \underset{\tilde{\mathbf{x}} \in \mathcal{S}^*(\theta_i, \mathbf{x}_k)}{\arg\max} \; ||\nabla_{\theta=\theta_i}\mathcal{L}(\tilde{\mathbf{x}}, \theta)|\,|_2, \tag{11}$$

where $\Phi(\theta, \boldsymbol{\gamma})$ is the first-order stochastic gradient descent (SGD) algorithm used to update θ as per descent direction $\boldsymbol{\gamma}$, e.g. in standard SGD, $\Phi(\theta, \gamma) = \theta + \beta\boldsymbol{\gamma}$, where β is the step-size (learning rate). Further $S^*(\theta_i, \mathbf{x}_k)$ represents the set of all maximizers of the robust loss,

$$S^*(\theta, \mathbf{x}, \mathcal{G}) = \underset{\substack{\tilde{\mathbf{x}}: \\ \mathcal{G}(\mathbf{x}, \tilde{\mathbf{x}},) \leq \epsilon,\ \tilde{\mathbf{x}} \in \mathcal{A}}}{\arg\max} \; \mathcal{L}(\tilde{\mathbf{x}}, \theta). \tag{12}$$

This set of (robust loss) maximizers, $S^*(\theta, \mathbf{x}, \mathcal{G})$ can theoretically be infinite. However, if assume we have access to a finite set with M adversarial examples, such that they define,

$$S^{*(M)}(\theta, \mathbf{x}) = \{\tilde{\mathbf{x}}^{(1)}, \ldots, \tilde{\mathbf{x}}^{(M)}\}, \tag{13}$$

then Latorre et al. (2023) propose an efficient algorithm termed, Danskin's Descent Direction (DDi), that provides a method to approximate the steepest direction, $\boldsymbol{\gamma}^*$ as though as if we are still selecting from the infinite set S^* [6], despite only having access to $S^{*(M)}$. The optimization problem over an infinite set in Equation 11 can be solved by finding an optimal linear combination, $\boldsymbol{\alpha} \in \triangle^M$ of the gradients of the loss, $\nabla_\theta g$ for each different adversarial example. Note that $\triangle^M$ defines the M-dimensional simplex (on which $\boldsymbol{\alpha}$ lies). If we let $\nabla_\theta g(\theta, S^{*(M)}_{1:K}(\theta))$ be the matrix with columns $\nabla_\theta g(\mathbf{x}_{1:K}, \theta_i, \tilde{\mathbf{x}}^{(m)}_{1:K}))$ for $m = 1, \ldots, M$, then

$$\boldsymbol{\gamma}^* = -\frac{\nabla_\theta g(\theta, S^{*(M)}_{1:K}(\theta))\boldsymbol{\alpha}^*}{||\nabla_\theta g(\theta, S^{*(M)}_{1:K}(\theta))\boldsymbol{\alpha}^*||_2},$$
$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha} \in \triangle^M}{\arg\min} ||\nabla_\theta g(\theta, S^{*(M)}_{1:K}(\theta))\boldsymbol{\alpha}||_2^2. \tag{14}$$

A.2 DDi-AT for NLP classification

The challenge with NLP is that generating strong textual adversarial examples as per Equation 13 can be extremely slow. Hence to increase speed, we generate adversarial examples in the token embedding space, such that we follow Equation 14, but adapt Equation 11 to,

$$g(\mathbf{x}_{1:K}, \theta_i, \hat{\tilde{\mathbf{h}}}_{1:K}) = \frac{1}{K}\sum_k \mathcal{L}(\hat{\tilde{\mathbf{h}}}_k, \theta_i),$$
$$\hat{\tilde{\mathbf{h}}}_k = \underset{\tilde{\mathbf{h}} \in \mathcal{S}^*(\theta_i, \mathbf{h}_k)}{\arg\max} \left||\nabla_{\theta=\theta_i}\mathcal{L}(\tilde{\mathbf{h}}, \theta)\right||_2, \tag{15}$$

where $\mathbf{h}_k = \{\mathbf{h}_{k,1}, \ldots, \mathbf{h}_{k,L}\}$ represents the sequence of token embeddings for tokens $\mathbf{x}_k = \{\mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,L}\}$. We can create our proxy finite set of maximizers, $S^{*(M)}$ (Equation 13) by using a computer-vision style Projected Gradient Descent (PGD) attack (Madry et al., 2019) in each token embedding space with initialisations of the PGD attack at different points to create multiple adversarial examples,

$$S^{*(M)}(\theta, \mathbf{h}) = \{\text{PGD}^{(1)}(\theta, \mathbf{h}), \ldots, \text{PGD}^{(M)}(\theta, \mathbf{h}), \}. \tag{16}$$

In this work we refer to DDi gradients applied to PGD AT as, *DDi-AT*.

A.3 Gradient Normalization and Overconfidence

It is shown in Table 1 that the use of the DDi gradients with the PGD AT approach (ddi-at) gives rise

[6] Theorem 3 in the paper justifies the conditions to certify that the approximation is the steepest descent direction

to a highly overconfident model, which is responsible for the IOR. This section aims to determine the route cause of this overconfidence in the DDi gradient update algorithm. Equation 11 indicates that in the DDi gradient update algorithm global gradient normalization is applied. Note that this is different to standard training algorithms where either no normalization is applied or gradient clipping is used where global gradient normalization is only applied if the global gradient norm is larger than a threshold (Pascanu et al., 2012). Table 5 demonstrates that the use of the global gradient normalization in DDi-AT is responsible for the overconfidence and thus IOR. Interestingly, Table 6 reveals that gradient normalization can also induce overconfidence for the standardly trained *baseline* model.

Normalization	**clean**	$\bar{P}(\hat{c}\|\mathbf{x}_{\text{clean}})$	$\bar{P}(\hat{c}\|\mathbf{x}_{\text{adv}})$
gradient norm	$87.90_{0.49}$	$99.97_{0.03}$	$99.91_{0.01}$
gradient clipping	$88.28_{0.68}$	$97.16_{0.30}$	$86.12_{0.72}$
none	$88.20_{0.55}$	$96.98_{0.42}$	$86.16_{0.66}$

Table 5: Model Confidence on clean and adversarial (pwws) examples for DDi-AT model with different forms of gradient normalization in the DDi gradient update step. Rotten Tomatoes dataset, DeBERTa model.

Normalization	**clean**	$\bar{P}(\hat{c}\|\mathbf{x}_{\text{clean}})$	$\bar{P}(\hat{c}\|\mathbf{x}_{\text{adv}})$
gradient norm	$88.56_{0.19}$	$99.96_{0.04}$	$99.93_{0.02}$
gradient clipping	$88.94_{0.31}$	$97.02_{0.29}$	$86.74_{0.84}$
none	$88.96_{0.30}$	$97.08_{0.26}$	$86.04_{0.68}$

Table 6: Model Confidence on clean and adversarial (pwws) examples for *baseline* model with different forms of gradient normalization in training. Rotten Tomatoes dataset, DeBERTa model.

B Dataset Descriptions

We conduct experiments across six standard NLP classification datasets to ensure our findings are robust (statistics summarised in Table 7). Rotten Tomatoes (*rt*; Pang and Lee, 2005) is a binary sentiment classification task for movie reviews. The Emotion Dataset (*emotion*; Saravia et al., 2018) categorizes Twitter tweets into one of six emotions: love, joy, surprise, fear, sadness or anger. The remaining three datasets are sourced from the the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019a).[7] The Corpus of Linguistic Acceptability (*cola*) dataset comprises English acceptability judgments sourced from books and journal articles on linguistic theory. Each instance consists of a word sequence annotated to indicate if it is grammatically correct. The Question-answering NLI (*qnli*) dataset assesses the task of sentence pair classification, where one sentence is a question and the other a context. The goal is to ascertain whether the context sentence contains the answer to the question. The Microsoft Research Paraphrase Corpus (*mrpc*) consists of pairs of sentences automatically extracted from online news sources. Human annotations identify if the sentences in each pair are semantically equivalent. Finally we consider the popular AGNews dataset (Zhang et al., 2015), consisting of articles from 2000 news sources classified into one of four topics: business, sci/tech, world or sports. There are a combined 120,000 training samples and 7600 test samples.

Dataset	#classes	Train	Validation	Test
rt	2	8.53k	1.07k	1.07k
emotion	6	16k	2k	2k
cola	2	8.55k	1.04k	1.06k
qnli	2	105k	5.46k	5.46k
mrpc	2	3.67k	408	1.73k
agnews	4	96k	24k	7.6k

Table 7: Dataset statistics

C Generative LLMs

With the advent of powerful generative Large Language Models, their use has become increasingly ubiquitous. However, we found that such popular generative models were not suitable in this work for the following reasons:

1. The encoder-only models used in this work are state-of-the-art when fine-tuned on classification tasks. Some recent studies (Periti et al., 2024; Zhong et al., 2023) have also found that often (fine-tuned) encoder-only models can be better than the popular generative LLMs for specific classification tasks. We show a comparison of performance in Table 8 below to a SOTA generative LLM (with zero-shot and few-shot prompting). We also present a comparison of the performance from Zhong

[7] For datasets where the provided test set is not labeled, we used the validation set.

et al. (2023) between BERT-based encoder models and ChatGPT (GPT3.5) on some of the datasets covered in our work (Table 9).

2. As a consequence of the competitive if not superior performance of finetuned encoder-based models on classification tasks, in many industry applications of NLP, such models (BERT-based models) are used extensively due to being light-weight (far fewer parameters than popular generative LLMs), cost-effective and still able to perform extremely well at many classification tasks.

3. The adversarial attack and defence literature that we are contributing to focuses on encoder models. Therefore, matching their setting allows us to build upon existing attacks and defences.

Model	#	rt	agnews	cola	mrpc	qnli
DeBERTa-base	110M	88.96	93.75	83.70	87.46	93.17
Mistral-7B-instruct-0.2 (0-shot)	7B	86.47	92.32	60.25	67.15	83.11
Mistral-7B-instruct-0.2 (5-shot)	7B	88.92	94.01	78.57	76.21	89.24

Table 8: Comparison of model performance of DeBERTa (used in this paper) with a popular generative LLM, Mistral (Jiang et al., 2023). # is the number of model parameters.

Model	#	cola	mrpc	qnli
BERT-base	110M	56.4	90.0	84.0
RoBERTa-base	110M	61.8	90.0	92.0
ChatGPT (0 shot)	unk	56.0	66.0	84.0
ChatGPT (1 shot)	unk	52.0	66.0	84.0
ChatGPT (5 shot)	unk	60.2	76.0	88.0
ChatGPT-CoT	unk	64.5	78.0	86.0

Table 9: Comparison of encoder-only models and generative LLMs as given in Zhong et al. (2023). # is the number of model parameters.

D Hyperparameter selection

We train the Transformer *baseline* models using standard hyper-parameter settings (He et al., 2020): initial learning rate of $1e-5$; batch size of 8; total of 5 epochs; 0 warm-up steps;[8] ADAMW optimizer, with a weight decay of 0.01 and parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-8$.

The Adversarial Training (AT) baseline approaches are trained with the same hyperparameters as for the *baseline* model and AT specific hyperparameters are as described in Li et al. (2021b). The default hyperparameters for each baseline (pgd, ascc and freelb) are: 5 adversarial iterations; adversarial learning rate of 0.03; adversarial initialisation magnitude of 0.05; adversarial maximum norm of 1.0; adversarial norm type of l2; α for ascc is 10.0; and β for ascc is 40.0. For DDi-AT, DDi gradients are applied to the PGD AT approach, with $M = 3$ gradients and $K = 3$ PGD iteration steps.

D.1 DDi-AT Ablation

The main results report DDi-AT results for DDi gradients applied to PGD AT with $K = 3$ PGD steps to find each adversarial example (in the embedding space) during training and $M = 3$ adversarial examples (refer to Section A.2). Table 10 gives the impact on adversarial accuracy (with and with out adversarial temperature calibration) of varying K and M. It appears that with greater iteration steps, K, the model presents a smaller IOR and a greater true robustness as the robustness accuracy does not degrade as much after calibration.

M	K	**Adv**	**clean**	**pwws**	**dg**
3	3	-	87.90 ±0.49	61.07 ±0.99	66.73 ±1.01
		cal	87.90 ±0.49	23.08 ±1.96	22.89 ±3.38
3	5	-	87.87 ±0.57	55.53 ±10.10	61.73 ±10.06
		cal	87.87 ±0.57	31.08 ±4.61	32.90 ±6.31
3	7	-	88.12 ±0.11	40.06 ±12.24	44.50 ±15.79
		cal	88.12 ±0.11	31.21 ±1.26	30.93 ±0.61
5	5	-	87.65 ±1.17	50.59 ±21.23	54.00 ±26.22
		cal	87.65 ±1.17	28.08 ±2.05	27.95 ±4.29
5	7	-	88.15 ±0.38	31.68 ±2.96	34.96 ±4.79
		cal	88.15 ±0.38	29.92 ±1.17	31.61 ±0.84

Table 10: Ablation: DDi-AT with M PGD adversarial examples, with each PGD adversarial example search during training using K iteration steps.

[8] We follow TextDefender (Li et al., 2021a) in setting no warm-up steps. Further, empirically validation accuracy remained the same with warm-up of 50 and 100 steps.

E Further Experiments for Illusion of Robustness and Mitigation

E.1 Experiments on Other Datasets

Equivalent results are presented for Twitter Emotions (6 emotion classes) in Table 11 and for the AGNews dataset (4 news classes) in Table 12.

Method	clean	bae	tf	pwws	dg
baseline	93.13 ±0.24	30.17 ±0.85	5.77 ±0.55	11.80 ±2.01	8.32 ±2.98
↓conf (§3.1)	93.13 ±0.24	29.63 ±0.80	6.78 ±0.58	15.22 ±1.55	14.68 ±3.01
↑conf (§3.1)	93.13 ±0.24	30.62 ±0.76	16.62 ±0.51	28.85 ±1.01	31.03 ±2.07
ddi-at (§3.2)	93.40 ±0.18	27.92 ±1.23	9.90 ±0.79	18.57 ±0.67	18.17 ±1.65
dg-aug	92.58 ±0.11	31.52 ±2.82	4.68 ±0.25	9.33 ±0.11	29.45 ±0.64
pgd	93.48 ±0.03	28.83 ±0.43	4.88 ±1.24	9.95 ±0.69	5.45 ±1.08
ascc	91.15 ±0.57	34.65 ±0.23	4.60 ±1.05	12.15 ±0.22	11.28 ±1.40
freelb	93.67 ±0.23	29.15 ±1.00	4.93 ±1.25	10.15 ±0.30	5.48 ±0.73

Table 11: **Twitter:** Extreme confidence systems compared to standard AT methods on out-of-the-box adversarial attacks.

Method	clean	bae	tf	pwws	dg
baseline	93.75 ±0.25	78.46 ±0.51	31.63 ±1.11	42.25 ±2.93	46.21 ±1.31
↓conf (§3.1)	93.75 ±0.25	81.08 ±0.51	59.17 ±0.19	70.79 ±2.24	75.71 ±1.06
↑conf (§3.1)	93.75 ±0.25	85.71 ±0.80	84.79 ±0.89	88.21 ±0.36	88.17 ±0.31
ddi-at (§3.2)	94.25 ±0.33	88.00 ±0.75	88.08 ±1.00	88.96 ±0.36	89.25 ±0.13
dg-aug	94.13 ±0.43	74.58 ±1.63	33.92 ±0.19	50.33 ±1.25	56.38 ±0.38
pgd	94.00 ±0.50	85.13 ±0.50	45.86 ±1.27	59.58 ±0.95	57.00 ±1.44
ascc	94.03 ±0.46	83.19 ±0.87	49.80 ±1.95	54.04 ±1.86	58.70 ±1.32
freelb	93.58 ±0.07	83.46 ±0.71	44.13 ±0.66	58.13 ±1.73	54.25 ±2.05

Table 12: **AGNews:** Extreme confidence systems compared to standard AT methods on out-of-the-box adversarial attacks. _*Evaluation on 1000 samples._

E.2 Experiments on Other Models

The *illusion of robustness* is presented for an overconfident, underconfident and DDi-AT *DeBERTa* model in the main paper in Table 2. The same trends are observed for other popular Transformer-encoder (*base*) models: RoBERTa (Table 13); and BERT (Table 14).

Method	clean	bae	tf	pwws	dg
baseline	88.27 ±0.47	32.46 ±0.74	17.01 ±0.72	21.23 ±0.05	24.30 ±1.71
↓conf	88.27 ±0.47	31.77 ±0.33	20.42 ±1.27	24.92 ±1.43	32.99 ±1.33
↑conf	88.27 ±0.47	37.65 ±0.76	53.63 ±0.94	58.66 ±0.61	66.32 ±0.92
ddi-at	88.06 ±0.62	36.24 ±0.85	50.84 ±0.41	54.85 ±1.25	62.76 ±1.27

Table 13: **RoBERTa** Model: Robustness of Mis-calibrated systems.

Method	clean	bae	tf	pwws	dg
baseline	85.08 ±0.50	30.52 ±0.76	21.01 ±0.32	21.20 ±0.34	23.14 ±2.14
↓conf	85.08 ±0.50	29.74 ±0.19	20.95 ±0.53	24.58 ±1.36	30.64 ±0.24
↑conf	85.08 ±0.50	35.08 ±1.11	45.84 ±0.85	53.25 ±1.37	57.50 ±2.06
ddi-at	85.55 ±0.43	36.80 ±0.29	48.09 ±0.69	51.50 ±1.04	56.60 ±1.16

Table 14: **BERT** Model: Robustness of Mis-calibrated systems.

E.3 Transferability Defence against IOR

One might even argue that to expose an IOR, it is unnecessary for an adversary to modify the model with adversarial temperature scaling to find adversarial examples. Instead, adversarial examples can be found for another model (e.g., *baseline*) and transferred to the target model. This follows from Demontis et al. (2018) where it is shown that similar architectures can be susceptible to the same adversarial examples. We test this hypothesis. It is clear from Table 15 that although the transfer attack from *baseline* to *ddi-at* is effective in reducing the adversarial accuracy, it is unable to bring the adversarial accuracy down to the same values as *baseline*, as achieved by our proposed temperature scaling approaches in Table 3.

source	target	clean	bae	tf	pwws	dg
baseline	baseline	88.96 ±0.30	31.39 ±1.20	17.82 ±0.49	20.42 ±0.62	20.11 ±0.94
ddi-at	ddi-at	87.90 ±0.49	39.18 ±0.75	56.54 ±1.67	61.07 ±0.99	66.73 ±1.01
baseline	ddi-at	87.90 ±0.49	48.91 ±0.60	52.47 ±1.15	50.00 ±1.64	48.53 ±0.99

Table 15: Transferability: adversarial examples for each attack method are generated for the source model and adversarial accuracy (%) is given for the target model.

E.4 Calibration Error

In Table 16 we verify that the calibration approaches are effective in calibrating the models. We report the metrics: Expected Calibration Error (ECE) and Maximum Calibration Error (MCE).

Method	ECE	MCE	$\bar{P}(\hat{c}\|\mathbf{x}_{\text{clean}})$	$\bar{P}(\hat{c}\|\mathbf{x}_{\text{adv}})$
baseline	48.82 ±0.62	51.98 ±1.15	97.08 ±0.26	86.04 ±0.68
↓conf	38.96* ±0.30	38.96* ±0.30	50.00007 ±0.00	50.00004 ±0.00
+cal	38.96* ±0.30	38.96* ±0.30	50.00004 ±0.00	50.00002 ±0.00
↑conf	51.31 ±1.03	62.62 ±11.8	99.98 ±0.02	99.95 ±0.01
+cal	42.30 ±0.91	48.28 ±1.04	90.36 ±0.45	75.88 ±0.58
ddi-at	52.41 ±0.57	74.87 ±20.97	99.97 ±0.03	99.91 ±0.05
+cal	42.60 ±0.58	62.73 ±18.36	90.13 ±0.11	87.54 ±0.80

Table 16: Calibration Error and Average Predicted Confidence (on clean and adv-pwws). N.B. baseline is across 3 seeds. *off-the-shelf calibration error computation fails here as all confidences very close to 50%, so manual computation of CE here: *accuracy - 50%*.

E.5 IOR in AT Approaches

The main results demonstrate that highly miscalibrated systems have an *illusion of robustness* (IOR), where an adversary's temperature calibration can mitigate this illusion of robustness. Considering the rotten tomatoes dataset and the DeBERTa model, Table 17 demonstrates that standard AT approaches considered in this work can also suffer from the IOR, when global gradient normalization is included in the training algorithm (Note that Table 6 shows that gradient normalization can be a source of model overonfidence). Nevertheless, Table 18 demonstrates that when global gradient normalization is excluded from the training algorithm, the baseline AT approaches considered in this work no longer present IORs as calibration does not degrade their adversarial accuracy.

Method	Adv	clean	bae	tf	pwws	dg
pgd*	-	88.59 ±0.64	39.94 ±0.55	58.02 ±1.04	64.45 ±0.77	67.02 ±0.83
	cal	88.59 ±0.64	33.71 ±0.20	17.73 ±0.86	25.20 ±1.80	25.74 ±1.46
ascc*	-	87.77 ±0.36	40.01 ±0.69	54.32 ±1.57	63.99 ±0.86	67.43 ±0.93
	cal	87.77 ±0.36	33.61 ±0.64	15.13 ±2.17	23.50 ±0.77	26.80 ±2.11

Table 17: Baseline AT approach (PGD and ASCC results here) can also suffer from IOR (calibration reduces observed adversarial robustness) when global gradient normalization used in the training algorithm. The IOR was also observed for dg-aug and freelb AT schemes.

Method	Adv	clean	bae	tf	pwws	dg
baseline	-	88.96 ±0.30	31.39 ±1.20	17.82 ±0.49	20.42 ±0.62	20.11 ±0.94
	cal	88.96 ±0.30	31.39 ±1.20	17.80 ±0.51	20.46 ±0.66	20.05 ±0.88
dg-aug	-	87.12 ±0.39	34.74 ±1.59	22.36 ±1.83	26.11 ±2.57	37.43 ±0.75
	cal	87.12 ±0.39	34.74 ±1.59	22.36 ±1.81	25.98 ±2.32	37.45 ±0.74
pgd	-	88.24 ±0.73	33.65 ±0.57	19.92 ±0.47	26.70 ±0.87	26.05 ±0.61
	cal	88.24 ±0.73	33.65 ±0.57	19.90 ±0.46	26.74 ±0.90	26.10 ±0.54
ascc	-	87.77 ±0.36	33.61 ±0.64	15.13 ±2.17	23.50 ±0.77	26.80 ±2.11
	cal	87.77 ±0.36	33.60 ±0.63	15.10 ±2.19	23.49 ±0.79	26.75 ±2.03
freelb	-	88.74 ±0.32	32.52 ±0.52	19.51 ±1.70	24.55 ±0.70	24.52 ±0.73
	cal	88.74 ±0.32	88.74 ±0.32	19.50 ±1.72	24.35 ±0.55	24.54 ±0.75

Table 18: Baseline AT approach can be freed of the IOR when global gradient normalization is not used in the training algorithm.

E.6 Alternative Calibration Approaches

In the main results, temperature calibration was implemented to detect adversarial examples based on two central considerations: 1) Temperature calibration effectively facilitates the adversarial attack search, especially for obviously mis-calibrated models; and 2) Temperature calibration preserves the rank order of logits, thereby ensuring transferability of adversarial examples from the calibrated to the original uncalibrated model. To broaden the analytical scope, alternative calibration techniques are examined. The goal is to assess their potential in mitigating the disruption to the adversarial attack search processes and to determine the potency of the resulting adversarial examples on the uncalibrated model. Binning-based calibration is deemed unsuitable due to its intrinsic non-differentiability, which could prevent the adversarial search process. Hence, the multi-class version of Platt Scaling is explored as a viable calibration strategy and subsequently contrasted against the benchmark temperature calibration approach from the main results. The performance of the calibration results is shown in Table 19, where it is evident that the Platt scaling approach is far less stable than temperature calibration and can in fact excessively enhance the *illusion of robustness*.

For automatic calibration, standard training hyperparameters were employed. Specifically, the temperature calibration protocol was set at 5,000 iterations with a learning rate of 0.01. Similarly, the Platt scaling protocol was also designed for 5000 iterations with a learning rate of 0.01. A point to note for practical implementation: adversaries might need to refine calibrator hyperparameters to minimize the Expected Calibration Error (ECE) on a specified validation set. However, ECE determination is nuanced, largely due to its sensitivity to chosen bin widths, as highlighted in Table 16 for instances of underconfidence.

Method	**Adv**	**clean**	**bae**	**tf**	**pwws**	**dg**
baseline	-	88.96 ±0.30	31.39 ±1.20	17.82 ±0.49	20.42 ±0.62	20.11 ±0.94
↓conf	-	88.96 ±0.30	31.21 ±0.94	20.98 ±0.99	25.17 ±0.89	32.18 ±2.78
	temp	88.96 ±0.30	31.52 ±0.34	21.89 ±0.43	27.58 ±1.31	31.52 ±0.34
	platt	88.96 ±0.30	72.08 ±12.15	70.33 ±18.00	72.70 ±16.72	74.73 ±17.11
↑conf	-	88.96 ±0.30	37.71 ±1.18	54.35 ±0.73	59.29 ±0.62	65.60 ±1.81
	temp	88.96 ±0.30	31.39 ±1.20	17.82 ±0.49	20.45 ±0.74	21.64 ±1.46
	platt	88.96 ±0.30	37.21 ±3.73	34.55 ±17.90	37.46 ±19.70	41.09 ±19.59
ddi-at	-	87.90 ±0.49	39.18 ±0.75	56.54 ±1.67	61.07 ±0.99	66.73 ±1.01
	temp	87.90 ±0.49	31.80 ±0.57	18.36 ±3.01	23.08 ±1.96	22.89 ±3.38
	platt	87.90 ±0.49	43.34 ±19.42	38.77 ±32.23	42.25 ±31.66	42.72 ±32.72

Table 19: Adversarial mitigation of highly miscalibrated systems using different test-time calibration approaches.

E.7 Extreme Miscalibration Leads to Masked Gradients

Section 3 argues that for heavily miscalibrated systems, the 'gradients' of the output probabilities with respect to the input are extremely noisy. Therefore, of-the-shelf adversarial attack methods, that use these gradients to select which tokens in the input sequence to attack, receive noisy signals and fail to operate. In this section, we demonstrate that extreme miscalibration does indeed cause noisy gradients for off-the-shelf-adversarial attacks. Note that these noisy gradients are referred to as *obfuscated* gradients or gradient masking by Athalye et al. (2018).

We consider two systems: the standard *baseline* system from the main paper and the heavily miscalibrated, overconfident system, ↑conf in the main paper. Experiments are on the *rt* dataset and we consider specifically the PWWS attack and Textfooler attack. These off-the-shelf adversarial attack approach rank all tokens w_i in the input sequence $\mathbf{x}$ by their influence on the output of the model (N.B. this is considered an approximation for the gradient of the output with respect to each input token). The PWWS attack refers to this influence as *saliency*, whilst the Textfooler attack calls it *importance*. To assess the impact of heavy miscalibration on the rank ordering, Table 20 reports the Spearman Rank Correlation between the rank of all input tokens (in the first iteration of the attack) as per the two models: *baseline* and ↑conf. The average correlation and standard deviation are given over the entire dataset. The average rank correlation is 0.28 for PWWS and 0.29 for Textfooler, which is very low and demonstrates that by simply having heavy miscalibration there is a significant impact on the attack mechanism. Further, the standard deviation is also large, suggesting that for many input sequences, the correlation is even lower.

Attack	Rank Correlation
pwws	0.28 ±0.24
textfooler	0.29 ±0.26

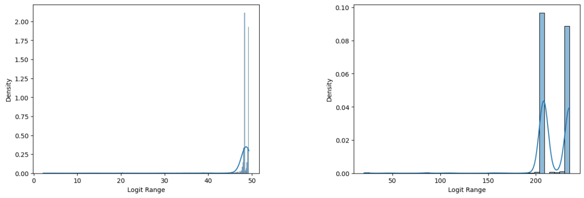
Table 20: Spearman Rank Correlation of input tokens' importance with (overonfident model) and without (*baseline* model) heavy miscalibration. The low rank correlation demonstrates that the token importance is strongly impacted by extreme confidence, which can explain the observed IOR for highly miscalibrated models.

F Further Experimental Results for High Temperature Training for Genuine Robustness

Here ST* will refer to a standardly trained *baseline* model (Equation 2), with gradient normalization during training whilst AT will refer to an adversarially trained model (Equation 3), as indicated by the naming convention given in Table 21.

	standard	adversarial
$T = 1$	ST	AT
High T	ST $\oplus$ T	AT $\oplus$ T

Table 21: Naming convention for experiments with different training objectives and high temperature training.



(a) $T = 1$ (b) $T = 100$

Figure 3: Probability Density (histogram plot) of predicted class logits' range (smallest logit subtracted from largest logit value) on *rt* test set with and without a high training temperature for the *baseline* ST DeBERTa model. The higher temperature training setting ($T = 100$) has a larger class logits' range, suggesting that an adversarial attack has to make a greater change in the logit space to be successful in changing the predicted class.

F.1 Class Margin Explanation

The success of high temperature training for adversarial robustness can perhaps be explained by considering the size of the class margin (Robey et al., 2023). A high temperature smooths the probability distribution across classes, such that the probabilities of the different classes are closer together. To minimize the cross entropy loss during the training, the model's parameters learn to compensate for this smoothing by pushing the logits of the different classes further apart (we see this in Figure 3, where the range of logits substantially increases with higher training temperatures). Intuitively, this can be viewed as increasing the distance to the class boundary in the logit space and thus making it more difficult for an adversarial attack to change the predicted class, giving rise to the observed increase in adversarial robustness. Future work will aim to rigorously understand and explain the observed robustness gains of training with a high temperature.

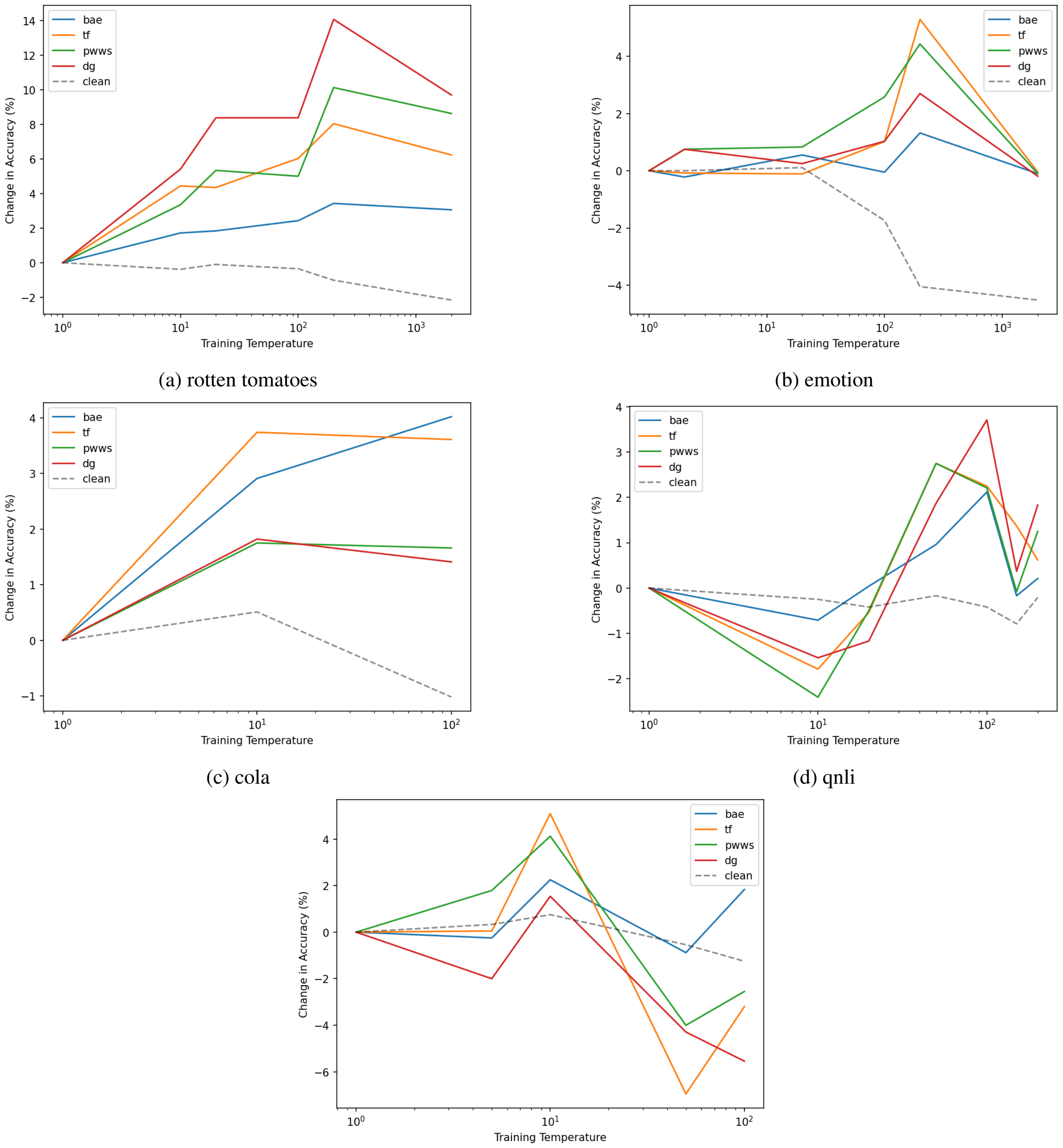
F.2 High Temperature Standard Training

For each dataset, Figure 4 presents the change in clean and adversarial accuracy of a standardly trained baseline *ST* model trained as per a standard training objective (Equation 2), with different temperatures T used during training. We present the detailed breakdown of the clean and adversarial accuracies for each training temperature, for each adversarial attack and each dataset, in Appendix F.5.

We first observe a general increase in adversarial accuracy (robustness) with the training temperature and then a decrease in the accuracy with extremely large training temperatures,[9] across all datasets. In some datasets (e.g., qnli and mrpc) there is a slight decrease in robustness before the sudden rise in robustness. Nevertheless, there exists a consistent robustness profile for each dataset, where robustness peaks at similar temperatures for all tested adversarial attack types (*bae, tf, pwws, and dg*). This is particularly useful, as a model developer, with access to only one form of adversarial attack, can tune the training temperature for optimal robustness on that specific attack form, yet be confident that the robustness gains will also transfer to the other unseen/unknown attack forms.

A further observation is that increasing temperature can lead to a small drop (between 1% and 4%) in clean accuracy. This is perhaps expected as the model can be viewed as being trained in a mode further from the optimal hyper-parameter setting. However, across all the datasets, the optimal temperature (aligned with the peak in adversarial accuracy) results in a maximal drop in clean accuracy of 1% (apart from for the emotion dataset). Given the gains in adversarial accuracy can be between 4% and 14%, this trade-off for clean accuracy can be acceptable. Further, a model developer can choose to operate at a different operating point, by selecting a training temperature that gives a smaller drop in clean accuracy (and settle for a less significant gain in the model robustness).

[9] The drop in robustness for extremely large training temperatures may be attributed to large temperatures excessively smoothing the predicted probability distribution during training, which makes it too challenging for the model to learn, as is reflected by the significant decrease in clean accuracy.



(a) rotten tomatoes

(b) emotion

(c) cola

(d) qnli

(e) mrpc

Figure 4: The use of a training temperature, T, is a simple adjustment in standard model training (ST), where the temperature parameter, T, is used to scale down predicted model logits. Higher training temperatures enhance model robustness against unseen adversarial attacks (*bae, tf, pwws, dg*) without requiring prior knowledge of these attack forms during training. This increased robustness is quantified by the absolute change in adversarial accuracy compared to the baseline $T = 1$ ST model.

F.3 High Temperature Adversarial Training

Here we explore the impact of combining the high temperature training approach with popular NLP AT methods. We consider four popular adversarial training approaches: dg-aug*, PGD*, FreeLB* and ASCC*. Table 22 and Table 23 give the baseline ST* results and AT results combined with the tem-

Method	clean	bae	tf	pwws	dg
baseline*	88.56 ±0.19	32.40 ±0.14	18.79 ±0.48	21.36 ±1.22	21.11 ±0.66
$\oplus T$	87.55 ±0.44	35.83 ±0.84	26.83 ±4.57	31.49 ±3.07	35.18 ±4.71
pgd*	88.59 ±0.64	33.71 ±0.20	17.73 ±0.86	25.20 ±1.80	25.74 ±1.46
$\oplus T$	87.77 ±0.43	34.77 ±0.33	24.55 ±1.76	31.46 ±1.08	31.77 ±2.64
freelb*	88.74 ±0.32	32.52 ±0.52	19.51 ±1.70	24.55 ±0.70	24.52 ±0.73
$\oplus T$	88.02 ±0.52	35.15 ±0.80	25.17 ±0.96	29.96 ±0.68	31.49 ±1.04
ascc*	87.77 ±0.36	33.61 ±0.64	15.13 ±2.17	23.50 ±0.77	26.80 ±2.11
$\oplus T$	86.36 ±0.80	34.93 ±1.12	27.36 ±0.72	30.93 ±1.38	33.46 ±1.65
dg-aug*	87.12 ±0.39	34.74 ±1.59	22.36 ±1.83	26.11 ±2.57	37.43 ±0.75
$\oplus T$	87.09 ±0.22	36.99 ±2.64	26.92 ±2.86	31.43 ±1.67	36.40 ±1.90

Table 22: Adversarial Training (AT) combined with a training temperature of $T = 200$ ($\oplus$ T). For each adversarial attack, the higher adversarial accuracy between the AT model and the AT $\oplus$ T model is underlined. In almost all cases, the higher training temperature improves adversarial accuracy. Test-time calibration (*cal*) is used to mitigate IOR. Dataset: Rotten Tomatoes.

Method	clean	bae	tf	pwws	dg
baseline*	93.13 ±0.24	30.17 ±0.85	5.77 ±0.55	11.80 ±2.01	8.32 ±2.98
$\oplus T$	92.83 ±0.89	32.10 ±0.95	6.42 ±1.58	12.68 ±1.20	8.45 ±1.37
pgd*	93.48 ±0.03	28.83 ±0.43	4.88 ±1.24	9.95 ±0.69	5.45 ±1.08
$\oplus T$	93.40 ±0.10	30.58 ±0.65	5.43 ±0.25	10.78 ±0.99	6.33 ±1.51
freelb*	93.67 ±0.23	29.15 ±1.00	4.93 ±1.25	10.15 ±0.30	5.48 ±0.73
$\oplus T$	93.72 ±0.10	30.23 ±0.53	5.58 ±0.03	10.78 ±0.96	5.23 ±0.98
ascc*	91.15 ±0.57	34.65 ±0.23	4.60 ±1.05	12.15 ±0.22	11.28 ±1.40
$\oplus T$	91.78 ±0.24	34.78 ±0.03	7.57 ±0.45	14.08 ±0.64	11.55 ±1.48
dg-aug*	92.58 ±0.11	31.52 ±2.82	4.68 ±0.25	9.33 ±0.11	29.45 ±0.64
$\oplus T$	91.98 ±0.13	31.88 ±0.85	5.38 ±0.28	9.40 ±0.87	23.63 ±1.26

Table 23: Adversarial Training (AT) combined with a training temperature of $T = 20$ ($\oplus$ T). For each adversarial attack column the higher adversarial accuracy between the AT model and the AT $\oplus$ T model is underlined. In almost all cases, a higher training temperature improves adversarial accuracy. Test-time calibration (*cal*) is used to mitigate IOR. Dataset: Emotion.

perature training approach on the *rt* and *emotion* datasets, respectively.

Although more significant for *rt* than *emotion*, for both datasets, combining with the high training temperature approach improves the adversarial accuracy for all adversarial attack forms (*bae*, *tf*, *pwws*, and *dg*) for the different adversarial training approaches PGD*, FreeLB*, and ASCC*. This demonstrates that high temperature training is complementary with such adversarial training approaches and thus consistently encourages a gain in robustness. Interestingly, we observe that for *dg-aug*, high temperature training is able to consistently improve adversarial accuracy for *bae*, *tf*, and *pwws* adversarial attacks, but can cause a drop in adversarial accuracy for the *dg* attack. It should be emphasized that *dg* in this context behaves as a *seen* attack form, as the training uses augmentation with *dg* adversarial examples, whilst the other attacks (*bae*, *tf*, and *pwws*) can be considered *unseen* attack forms that the model developer has no knowledge of during training. This suggests that for augmentation-based NLP adversarial training approaches, a high training temperature does not necessarily increase robustness to *seen* attack forms, but is successful in boosting robustness to *unseen* attack forms.

F.4 Transferability of High Temperature Training

It is shown that training with a high temperature leads to a consistent gain in adversarial robustness to unseen adversarial attack forms. However, an adversary may attempt to exploit attack *transferability* when looking to attack the target model trained with high temperature. To explore this notion of a transfer attack, with the *rt* dataset, Table 15 shows the impact of finding adversarial examples for the source baseline ST* model and assessing their efficacy on the target baseline ST* $\oplus$ T model. It is evident from the significant increase in the adversarial accuracy for all the attack forms (*bae*, *tf*, *pwws*, and *dg*), that a transfer attack is not able to degrade the observed robustness gains for models trained with high temperatures.

Source	Target	bae	tf	pwws	dg
baseline*	baseline*	31.39 ±1.20	17.82 ±0.49	20.42 ±0.62	20.11 ±0.94
$\oplus T$	$\oplus T$	35.83 ±0.84	26.83 ±4.57	31.49 ±3.07	35.18 ±4.71
baseline*	$\oplus T$	50.13 ±0.30	46.90 ±0.38	47.53 ±1.22	46.09 ±1.13

Table 24: Transferability: adversarial examples are found for the *source* model and evaluated on the *target* model on the *rt* test set. The results here demonstrate that the standard trained, high temperature ($\oplus T$) model's robustness gains relative to the baseline* model cannot be compromised by a transferability attack, i.e. the performance of the $\oplus T$ model are not degraded by adversarial examples generated from the baseline* model. Test-time calibration (*cal*) is used to mitigate IOR.

F.5 Detailed Performance Breakdown

Figure 4 presents the adversarial accuracy of baseline ST* models trained with different training temperatures. In this section, for reference, we provide the detailed breakdown (average across 3 seeds and standard deviation) of performances for the different training temperatures for each dataset: *rt* (Table 25), *emotion* (Table 26), *cola* (Table 27), *qnli* (Table 28), and *mrpc* (Table 29). These results are given for the DeBERTa model as in the main paper.

Temp	clean	bae	tf	pwws	dg
1	88.56 ±0.19	32.40 ±0.14	18.79 ±0.48	21.36 ±1.22	21.11 ±0.66
10	88.18 ±0.49	34.12 ±0.82	23.23 ±2.65	24.71 ±1.22	26.52 ±2.10
20	88.46 ±0.52	34.24 ±1.97	23.14 ±2.18	26.70 ±1.28	29.49 ±1.69
100	88.21 ±0.70	34.83 ±0.81	24.82 ±3.79	26.36 ±2.86	29.49 ±4.36
200	87.55 ±0.44	35.83 ±0.84	26.83 ±4.57	31.49 ±3.07	35.18 ±4.71
2000	86.40 ±0.89	35.46 ±1.79	25.02 ±0.76	29.99 ±1.38	30.81 ±1.05

Table 25: **rt:** The use of a training temperature, T, is a simple adjustment in standard baseline model training (baseline*), where the temperature parameter, T, is used to scale down predicted model logits. Higher training temperatures enhance model robustness against unseen adversarial attacks (*bae, tf, pwws, dg*) without requiring prior knowledge of these attack forms during training. Results here report the clean and adversarial accuracy. Test-time calibration (*cal*) is used to mitigate IOR.

Temp	clean	bae	tf	pwws	dg
1	92.72 ±0.10	31.55 ±0.20	6.53 ±1.30	11.85 ±1.31	8.20 ±1.22
2	92.72 ±0.36	31.33 ±0.83	6.45 ±1.66	12.60 ±2.26	8.95 ±2.43
20	92.83 ±0.89	32.10 ±0.95	6.42 ±1.58	12.68 ±1.20	8.45 ±1.37
100	90.98 ±0.15	31.50 ±0.79	7.55 ±0.10	14.43 ±0.74	9.23 ±1.87
200	85.67 ±0.24	32.87 ±0.47	11.82 ±0.64	16.28 ±0.66	10.90 ±1.44

Table 26: **emotion:** The use of a training temperature, T, is a simple adjustment in standard baseline model training (baseline*), where the temperature parameter, T, is used to scale down predicted model logits. Higher training temperatures enhance model robustness against unseen adversarial attacks (*bae, tf, pwws, dg*) without requiring prior knowledge of these attack forms during training. Results here report the clean and adversarial accuracy. Test-time calibration (*cal*) is used to mitigate IOR.

In Table 30, we further include results on a 6th dataset AGNews (Zhang et al., 2015), where there are four news classes, 96k training samples, 24k validation samples and 7.6k test samples. For this dataset, it can be observed that a high training tem-

Temp	clean	bae	tf	pwws	dg
1	83.70 ±0.53	3.39 ±0.59	5.43 ±0.43	10.23 ±0.73	11.63 ±1.49
10	84.21 ±0.72	6.30 ±0.83	9.17 ±0.48	11.98 ±0.42	13.45 ±1.75
100	82.68 ±0.91	7.41 ±2.34	9.04 ±3.22	11.89 ±1.08	13.04 ±4.96

Table 27: **cola:** The use of a training temperature, T, is a simple adjustment in standard model training (baseline*), where the temperature parameter, T, is used to scale down predicted model logits. Higher training temperatures enhance model robustness against unseen adversarial attacks (*bae, tf, pwws, dg*) without requiring prior knowledge of these attack forms during training. Results here report the clean and adversarial accuracy. Test-time calibration (*cal*) is used to mitigate IOR.

Temp	clean	bae	tf	pwws	dg
1	93.17 ±0.26	35.71 ±1.88	20.71 ±3.17	19.79 ±2.38	17.92 ±4.39
10	92.92 ±0.94	35.00 ±0.66	18.92 ±1.56	17.38 ±1.02	16.38 ±2.76
20	92.75 ±0.66	35.75 ±0.38	20.17 ±1.54	19.29 ±0.90	16.75 ±1.19
50	93.00 ±0.75	36.67 ±1.39	23.46 ±1.70	22.54 ±0.26	19.79 ±1.56
100	92.75 ±0.33	37.83 ±1.19	22.96 ±0.38	22.00 ±1.44	21.63 ±1.11
150	92.38 ±0.70	35.54 ±2.00	22.08 ±2.12	19.71 ±2.32	18.29 ±3.59
200	92.96 ±0.47	35.92 ±1.21	21.33 ±3.54	21.04 ±2.48	19.75 ±3.06

Table 28: **qnli:** The use of a training temperature, T, is a simple adjustment in standard model training (baseline*), where the temperature parameter, T, is used to scale down predicted model logits. Higher training temperatures enhance model robustness against unseen adversarial attacks (*bae, tf, pwws, dg*) without requiring prior knowledge of these attack forms during training. Results here report the clean and adversarial accuracy. Test-time calibration (*cal*) is used to mitigate IOR.

Temp	clean	bae	tf	pwws	dg
1	87.46 ±0.26	46.42 ±1.94	38.83 ±3.92	28.63 ±3.25	33.50 ±6.11
5	87.79 ±0.44	46.17 ±3.05	38.88 ±4.58	30.42 ±2.20	31.50 ±1.11
10	88.21 ±0.31	48.67 ±3.62	43.92 ±5.63	32.75 ±3.69	35.04 ±5.03
50	86.92 ±0.29	45.54 ±4.15	31.88 ±8.15	24.63 ±5.00	29.21 ±7.22
100	86.21 ±0.94	48.25 ±2.58	35.63 ±6.39	26.08 ±5.59	27.96 ±4.74

Table 29: **mrpc:** The use of a training temperature, T, is a simple adjustment in standard model training (baseline*), where the temperature parameter, T, is used to scale down predicted model logits. Higher training temperatures enhance model robustness against unseen adversarial attacks (*bae, tf, pwws, dg*) without requiring prior knowledge of these attack forms during training. Results here report the clean and adversarial accuracy. Test-time calibration (*cal*) is used to mitigate IOR.

perature is not a successful method unless a fraction of the dataset (10k training samples) is used during training. Future work is necessary to understand the nature of this specific dataset or other similar datasets that led to such a different behaviour for the temperature training approach.

Temp	clean	bae	tf	pwws	dg
1	93.88 ±0.22	81.50 ±0.25	29.46 ±0.19	43.00 ±2.19	39.08 ±2.89
1.5	93.75 ±0.13	80.92 ±0.51	29.08 ±3.26	42.13 ±5.20	38.58 ±3.19
2	93.92 ±0.07	80.04 ±0.56	25.00 ±3.80	40.38 ±3.19	38.54 ±5.69
20	93.83 ±0.36	79.25 ±1.02	23.50 ±2.07	37.58 ±2.60	34.58 ±4.56

Table 30: **agnews:** The use of a training temperature, T, is a simple adjustment in standard model training (baseline*), where the temperature parameter, T, is used to scale down predicted model logits. Higher training temperatures enhance model robustness against unseen adversarial attacks (*bae, tf, pwws, dg*) without requiring prior knowledge of these attack forms during training. Results here report the clean and adversarial accuracy. Test-time calibration (*cal*) is used to mitigate IOR.

Temp	clean	bae	tf	pwws	dg
1	93.17 ±0.38	78.00 ±0.54	32.33 ±2.32	42.08 ±0.47	40.54 ±2.53
10	92.08 ±0.40	79.00 ±0.66	38.33 ±2.89	50.42 ±2.09	46.54 ±0.63
20	92.46 ±0.19	77.92 ±0.69	38.33 ±2.63	49.21 ±2.89	45.67 ±1.61
100	92.13 ±0.38	77.50 ±0.00	30.33 ±2.81	41.46 ±0.76	40.38 ±2.34

Table 31: **agnews:** The use of a training temperature, T, is a simple adjustment in standard model training (ST*), where the temperature parameter, T, is used to scale down predicted model logits. Higher training temperatures enhance model robustness against unseen adversarial attacks (*bae, tf, pwws, dg*) without requiring prior knowledge of these attack forms during training. Results here report the clean and adversarial accuracy. Training with 10k samples - 1/10th of default agnews training set size. Test-time calibration (*cal*) is used to mitigate IOR.

F.6 Reproducing with Other Models

The main paper presents results using the DeBERTa model. Here we repeat the core experiments on other popular *baseline* models: BERT (Table 32) and RoBERTa (Table 33). The results here are presented for the *rt* dataset.

Temp	clean	bae	tf	pwws	dg
1	85.08 ±0.50	30.52 ±0.76	21.01 ±0.32	21.20 ±0.34	23.14 ±2.14
10	84.79 ±0.58	32.16 ±0.66	25.88 ±1.23	23.96 ±1.89	27.48 ±1.67
100	84.76 ±0.54	33.01 ±0.78	27.12 ±1.99	25.88 ±2.45	28.92 ±2.02

Table 32: **BERT:** The use of a training temperature, T, is a simple adjustment in standard model training (ST*), where the temperature parameter, T, is used to scale down predicted model logits. Higher training temperatures enhance model robustness against unseen adversarial attacks (*bae, tf, pwws, dg*) without requiring prior knowledge of these attack forms during training. Results here report the clean and adversarial accuracy. Test-time calibration (*cal*) is used to mitigate IOR. Result for *rt* dataset.

Temp	clean	bae	tf	pwws	dg
1	88.27 ±0.47	32.46 ±0.74	17.01 ±0.72	21.23 ±0.05	24.30 ±1.71
10	88.25 ±0.65	33.17 ±0.86	21.96 ±1.86	24.32 ±1.15	28.85 ±3.02
100	88.26 ±0.72	33.55 ±0.92	23.20 ±2.04	26.03 ±2.12	29.66 ±3.55

Table 33: **RoBERTa:** The use of a training temperature, T, is a simple adjustment in standard model training (ST*), where the temperature parameter, T, is used to scale down predicted model logits. Higher training temperatures enhance model robustness against unseen adversarial attacks (*bae, tf, pwws, dg*) without requiring prior knowledge of these attack forms during training. Results here report the clean and adversarial accuracy. Test-time calibration (*cal*) is used to mitigate IOR. Result for *rt* dataset.

F.7 High Temperature Training GradNorm Ablation

We find that including gradient normalization during training with a high temperature is beneficial in preventing a decrease in clean accuracy, whilst maintaining the gains in genuine adversarial robustness. This is demonstrated in the results in Table 34, where calibration is used at test-time to ensure there is no IOR.

	clean	bae	tf	pwws	dg
baseline $\oplus T$	85.12 ±0.20	35.71 ±1.77	26.85 ±5.72	31.26 ±5.14	35.82 ±6.20
baseline* $\oplus T$	87.55 ±0.44	35.83 ±0.84	26.83 ±4.57	31.49 ±3.07	35.18 ±4.71

Table 34: Gradnorm (*) Ablation. Results are presented for the baseline system for the *rt* dataset and the DeBERTa model. A high training temperature of $T = 200$ is used. Test-time calibration (*cal*) is used to mitigate IOR.

F.8 IOR From High Temperature Training

In the main paper we present high temperature training as an effective method to induce genuine adversarial robustness. Here we demonstrate the need to apply the test-time calibration methods of Section 4 to ensure there is no IOR. From Table 35, we see an approximately two-fold increase in adversarial accuracy when no calibration is applied before evaluation.

Method	**Adv.**	**clean**	**bae**	**tf**	**pwws**	**dg**
baseline* $\oplus$T	-	87.55 ±0.44	38.87 ±0.81	56.62 ±0.41	61.01 ±0.49	67.20 ±1.41
	cal	87.55 ±0.44	35.83 ±0.84	26.83 ±4.57	31.49 ±3.07	35.18 ±4.71

Table 35: Adversarial Accuracy of high training temperature method ($T = 200$) with test-time calibration: none (-) or temperature scaling (-). **rt** dataset on DeBERTa model.

F.9 Other Ablations

Augmentation based adversarial training approaches, such as dg-aug* in the main paper, have twice as many training steps (due to there being double the training set size). To match the standard training setting, in Table 36 we evaluated the training with high temperature approach combined with dg-aug* at half the number of training steps. Similarly, in Table 37 we consider the inverse setting, where we double the number of training iterations for the *baseline** model (in standard training), as well as linearly scaling the learning rate scheduler across the increased number of iterations.

Method	iters	clean	pwws	dg
baseline* $\oplus T$	default	87.55 ± 0.44	31.49 ± 3.07	35.18 ± 4.71
dg-aug* $\oplus T$	default	87.09 ± 0.22	31.43 ± 1.67	36.40 ± 1.90
	half	86.05 ± 0.44	37.02 ± 5.21	43.00 ± 2.36

Table 36: Matched number of iterations for baseline* and high temperature training with dg-aug* by halving the number of training steps for dg-aug*. Test-time calibration (*cal*) is used to mitigate IOR.

Method	Epochs	clean	bae	tf	pwws	dg
baseline*	5	88.56 ± 0.19	32.40 ± 0.14	18.79 ± 0.48	21.36 ± 1.22	21.11 ± 0.66
	10	88.34 ± 0.62	33.61 ± 0.52	18.76 ± 0.50	22.39 ± 0.61	23.45 ± 0.86
$\oplus T$	5	87.55 ± 0.44	35.83 ± 0.84	26.83 ± 4.57	31.49 ± 3.07	35.18 ± 4.71
	10	87.55 ± 0.33	34.43 ± 1.31	25.48 ± 2.16	30.11 ± 2.19	33.40 ± 4.60
dg-aug*	-	87.12 ± 0.39	34.74 ± 1.59	22.36 ± 1.83	26.11 ± 2.57	37.43 ± 0.75
$\oplus T$	-	87.09 ± 0.22	36.99 ± 2.64	26.92 ± 2.86	31.43 ± 1.67	36.40 ± 1.90

Table 37: Doubling training iterations for the baseline* model with scaled scheduler decay to match number of iterations in augmentation based AT. Test-time calibration (*cal*) is used to mitigate IOR.

G Algorithms

G.1 Method for Temperature Scaling Optimization to Mitigate IOR

```
def optimize_temp(self, factor=10):
'''return temperature optimized to be successful in dg attack'''

adv_temp = 1
acc = self.eval_attack(adv_temp)

# check whether we need to increase or decrease temp
test_temp_acc = self.eval_attack(adv_temp*factor)
if test_temp_acc < acc:
    left = adv_temp
    right = 1e6 # assumes this as a maximum
else:
    left = 1e-10 # 0 causes floating point errors
    right = adv_temp

# seach for optimal temp (minima adv acc) using Brent algorithm (faster convergence of golden section algorithm)
opt_temp = scipy.optimize.brent(self.eval_attack, brack=(left, 0.5*(left+right), right), maxiter=10)

return opt_temp
```

G.2 Base Class Definition with High Temperature Training for Genuine Robustness

```
class BaseClassifier(nn.Module):
    def __init__(self, model_name='bert-base-uncased', num_labels=2, pretrained=True, temperature=1):
        super().__init__()
        self.model_name = model_name
        self.temperature = temperature
        if pretrained:
            self.model = AutoModelForSequenceClassification.from_pretrained(model_name, num_labels=num_labels)
            self.tokenizer = AutoTokenizer.from_pretrained(model_name)
        else:
            config = AutoConfig.from_pretrained(model_name, num_labels=num_labels) # returns config and not pretrained weights
            self.model = AutoModelForSequenceClassification.from_config(config)
            self.tokenizer = AutoTokenizer.from_pretrained(model_name)
        self.config = AutoConfig.from_pretrained(model_name, num_labels=num_labels)

    def forward(self, input_ids=None, attention_mask=None, inputs_embeds=None):
        logits = self.model(input_ids, attention_mask=attention_mask, inputs_embeds=inputs_embeds)[0]
        logits = logits / self.temperature
        return logits
```