From Robustness to Improved Generalization and Calibration in Pre-trained Language Models

Josip Jukić Jan Šnajder
TakeLab, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia
{josip.jukic,jan.snajder}@fer.hr

Abstract

Enforcing representation smoothness in pre-trained language models (PLMs) through Jacobian and Hessian regularization provides an effective approach for enhancing both robustness and generalization. Although such regularization methods have proven effective in computer vision, their application in natural language processing, where PLM inputs are derived from a discrete domain, poses unique challenges. We introduce JacHess, a regularization approach for PLMs that minimizes the norms of the Jacobian and Hessian matrices in intermediate representations, using embeddings as substitutes for discrete token inputs. JacHess supports dual-mode regularization, alternating between fine-tuning with labeled data and regularization with unlabeled data. We evaluate JacHess on the GLUE benchmark and demonstrate that it consistently and significantly improves in-distribution generalization and enhances performance under domain shift. Across diverse PLMs, JacHess outperforms comparable representation-based regularization methods and unregularized fine-tuning, while also improving model calibration. Our findings, coupled with a computationally efficient estimator for the Jacobian and Hessian norms, position JacHess as a robust and widely applicable solution for enhancing PLM performance.

1 Introduction

Effective generalization, broadly understood as the capability to transfer learned representations, knowledge, and strategies from familiar contexts to new ones (Hupkes et al., 2023), stands out as a key goal for models in natural language processing (NLP) and extends to the broader domain of machine learning. At the heart of machine learning optimization is the principle of minimizing empirical risk, which acts as a surrogate for the true risk. This approach inevitably leads to a *generalization gap*—the discrepancy between the empirical and true risks. The size of this gap can be gauged by considering certain model properties. Notably, *robustness*, defined as the amount that the loss can vary with respect to changes in the inputs, offers an effective lens for understanding generalization (Vapnik, 1995; Deng et al., 2021; Zhang et al., 2021). Recently, Kawaguchi et al. (2022) provided tighter generalization bounds for robustness, a challenge that has persisted since robustness was first proposed as a tool for analyzing learning algorithms (Xu and Mannor, 2012). However, their bounds are data-dependent and thus remain undetermined until the training data is specified, leaving room for further exploration.

Robustness in neural networks is intrinsically linked to the concept of *representation smoothness* (Bubeck and Sellke, 2021). When representations are smooth, the outputs of the network vary minimally in response to small input changes, thereby maintaining a stable loss by controlling the network's geometric complexity (Dherin et al., 2022). Promoting representation smoothness not only aids in better generalization but also supports more reliable uncertainty quantification in the model's predictions (Rosca et al., 2020), a critical area where neural networks often fail. Typically, this failure manifests as overconfidence in the model's predictions, resulting in poor calibration with predicted probabilities misaligned with actual outcomes (Guo et al., 2017). Promoting representation smoothness thus emerges as a promising strategy for boosting generalization through robustness and improving uncertainty quantification in neural networks.

While traditional regularization methods such as weight decay (Loshchilov and Hutter, 2018) and dropout (Srivastava et al., 2014) enhance neural network generalization by promoting model simplicity and preventing overfitting, they do not ensure robustness against input variations. In contrast, more specialized approaches focus on

Transactions of the Association for Computational Linguistics, vol. 13, pp. 264–280, 2025. https://doi.org/10.1162/tacl_a_00739
Action Editor: Dani Yogatama. Submission batch: 5/2024; Revision batch: 11/2024; Published 3/2025.
© 2025 Association for Computational Linguistics. Distributed under a CC-BY 4.0 license.

manipulating the network's input-output **Jacobian** and **Hessian** matrices, involving first-order and second-order partial derivatives. Minimizing the norms of these matrices increases robustness by promoting representation smoothness. This effect can be understood by interpreting a neural network as a function. The concept of *Lipschitz continuity* reflects the smoothness of a function against input variations, representing the maximum rate of change in the function's output relative to a change in the input (Khromov and Singh, 2024). Reducing the network's Lipschitz constant thus reduces its sensitivity to input perturbation. While computing the exact Lipschitz constant is an NP-hard problem (Virmaux and Scaman, 2018), leveraging Jacobian and Hessian norms as proxies offers a practical way to emulate these effects. The norm-based regularization approach has proven highly successful in the field of computer vision, as evidenced by several studies (Czarnecki et al., 2017; Varga et al., 2018; Sokolić et al., 2017; Mustafa et al., 2020).

While regularization methods directly targeting robustness have effectively enhanced generalization in computer vision, none have been used in NLP, revealing a significant research gap. In particular, given the pivotal role of pre-trained language models (PLMs) in advancing NLP, there exists an untapped potential to leverage representation-based regularization methods to improve PLMs' generalization capabilities. However—unlike in computer vision—the discrete nature of the tokens processed by PLMs poses a significant barrier to this research direction in NLP. This issue echoes previous challenges encountered when methods proven effective in computer vision did not seamlessly transfer to NLP (e.g., the idea of generative adversarial networks [Goodfellow et al., 2020] that operate by applying slight continuous modifications to inputs).

In this paper, we adapt and expand upon the representation-based regularization techniques used in computer vision and apply them to NLP. The discrete nature of data in PLMs makes it challenging to enhance robustness to input variations, but we find a workaround by leveraging the **continuous embedding space** as an effective alternative. We introduce JacHess, a novel regularization approach that minimizes the norms of the Jacobian and Hessian matrices within PLM representations relative to their inputs. By targeting both Jacobian and Hessian norms, JacHess doubly enhances model robustness by reducing sensitivity to input changes and smoothing the curvature of representations. Typically, calculating these norms in high-dimensional spaces is compute-intensive, but we address this using a computationally efficient estimator. Our method applies regularization across the network's layers, promoting smoothness in intermediate representations. Moreover, JacHess employs a dual-mode strategy, cycling between iterative fine-tuning using labeled data and regularization with additional **unlabeled** data. While employing JacHess with training data significantly improves generalization in most cases, utilizing a separate set of unlabeled data for regularization—which is typically more readily available than labeled data—enhances the model's generalization capabilities even further. Additionally, we demonstrate that JacHess proves highly beneficial under domain shift, where models need to generalize across different domains while performing the same underlying task.

Given their growing importance, our evaluation concentrates on decoder-based models, including the OPT family and Llama 2, where we fine-tune the PLMs and evaluate them on the GLUE benchmark (Wang et al., 2018). We validate JacHess by examining PLM robustness against the perturbations in embedding space and its impact on handling corrupted token inputs. Subsequently, we assess the effectiveness of JacHess in improving models' predictive accuracy and calibration. Our results reveal that JacHess markedly outperforms standard unregularized fine-tuning and other Jacobian- and Hessian-based methods, achieving a 2% to 4.5% absolute increase in average GLUE scores across various PLMs. This enhancement in generalization is complemented by improved calibration of model predictions.

In summary, our work presents two significant contributions: (1) we empirically evaluate the enforcement of representation smoothness on PLMs through Jacobian- and Hessian-based regularization, an aspect that has thus far been overlooked in NLP, and (2) we introduce JacHess, a novel regularization approach that not only improves model generalization beyond the capabilities of standard fine-tuning and other regularization methods in the representation space but also offers more reliable uncertainty quantification through improved calibration and enhanced robustness to domain shift. Taken together, this work sets a new standard for

improving the generalization and calibration of PLMs in the evolving NLP landscape.[1]

2 Related Work

Generalization Enhancers. Applying regularization to PLMs is a common practice during training to improve generalization (Liu et al., 2023). Although increasing the amount of labeled data can straightforwardly improve generalization, the challenge lies in efficiently utilizing existing data or capitalizing on the plentiful supply of unlabeled data. Various strategies aim to address this challenge. Notably, Gururangan et al. (2020) proposed task-adaptive pre-training, which involves additionally training the model on the task-specific unlabeled set via the language modeling task. In a different approach, Yu et al. (2021) leveraged unlabeled data for generalization enhancement through weak supervision. Moreover, beyond the utilization of unlabeled data, the field has seen advancements in generalization through data augmentation methods (Okimura et al., 2022; Zhou et al., 2022; Wu et al., 2022).

Representation Smoothness. Examining the impact of input changes on a function's output is essential for enhancing the generalization capabilities of neural networks. Rosca et al. (2020) explored the concept of *model smoothness*, or representation smoothness in the context of deep neural networks. They advocate for implementing smoothness constraints with respect to inputs to improve generalization and provide reliable uncertainty quantification. While numerous theoretical approaches predominantly focus on Lipschitz-bounds to assess model sensitivity (Bartlett et al., 2017; Bubeck and Sellke, 2021; Wang and Manchester, 2023), it has been demonstrated that Lipschitz continuity can be effectively leveraged for regularization purposes (Gouk et al., 2021).

Representation-based Regularization. Drucker and Le Cun (1992) were the first to explore the effects of the input-output Jacobian matrix of neural networks. Since then, several iterations of this technique have been proposed (Sokolić et al., 2017; Ororbia II et al., 2017), primarily aimed at enhancing robustness against adversarial examples (Schmidt et al., 2018; Li et al., 2022). Extending this work, Varga et al. (2018) suggested that such robustness could also improve in-distribution generalization. Further developments by Hoffman et al. (2019) introduced an estimator for the Frobenius norm of the Jacobian matrix of logits with respect to inputs, avoiding the need to compute the entire Jacobian and significantly reducing resource demands. Extending the scope of regularization, Mustafa et al. (2020) proposed regularizing the Frobenius norm of the Hessian matrix, focusing on the scalar outputs of binary classification, a technique they termed Cross-Hölder regularization. In a related approach, Aghajanyan et al. (2021) introduced an adversarial-style method by adding noise directly to the inputs during fine-tuning. Known as robust fine-tuning with regularized representations (R3F), this approach draws on earlier works (Zhu et al., 2020; Jiang et al., 2020) and leverages trust-region theory to enhance model stability. By reducing the divergence between outputs generated from noisy and clean inputs, R3F promotes smoothness within the representation space.

Sharpness-aware Minimization. The proposed JacHess method improves generalization by enhancing smoothness within the representation space of a PLM, which is similar in spirit to sharpness-aware minimization (SAM; Foret et al., 2021). SAM is an optimization technique designed to enhance model generalization by locating parameters that reside in flatter regions of the loss landscape. Bahri et al. (2022) showed that SAM improves PLM generalization, particularly in transfer learning scenarios. Extending this concept, Sherborne et al. (2024) leveraged SAM for cross-lingual tasks, improving the transfer of pre-trained representations across languages.

In our work, we focus on promoting smoothness with respect to inputs by applying regularization directly within the representation spaces of PLMs. Specifically, we leverage Jacobian- and Hessian-based regularization techniques to enforce smoothness across intermediate representations at all PLM layers, as opposed to restricting regularization to the model's final outputs or logits. Furthermore, we push beyond traditional regularization techniques by exploring the application of regularization on separate, unlabeled data.

[1]Our code is available at `https://github.com/josipjukic/jachess`.

3 Representation-based Regularization

In this section, we outline our approach, starting with the theoretical background of Lipschitz continuity, a useful tool for assessing the stability and smoothness of neural networks. Next, we address computational challenges and propose strategies to mitigate them. Finally, we introduce our regularization technique, JACHESS, which leverages the continuous embedding space of PLMs to employ representation-based regularization with respect to the inputs.

3.1 Lipschitz Continuity

Neural networks are commonly understood as functions that map inputs from an n-dimensional space to an m-dimensional space, formally represented as $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$. The characteristics of these functions are closely linked to the network's reliability and generalization capabilities. One way to characterize the stability of such functions is through *Lipschitz continuity*, which provides a formal framework for assessing how sensitive a function's output is to small changes in the input (Rudin, 1964). Intuitively, when a neural network exhibits large variations in output due to minor changes in input, it risks overfitting or producing unstable predictions. Lipschitz continuity quantifies the bound of how much the output can change in response to input variations, offering a measure of the network's robustness (Khromov and Singh, 2024).

Definition 1 (L-Lipschitz continuity)**.** *A function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is termed L-Lipschitz continuous if there is a real constant $L \geq 0$ such that:*

$$\|f(\mathbf{x}) - f(\mathbf{x}')\| \leq L\|\mathbf{x} - \mathbf{x}'\|, \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n.$$

Importantly, there is a link between the Lipschitz constant and the **Jacobian** matrix, which is often exploited when estimating the smoothness of functions implemented by neural networks. Let $\mathbf{J}_f(\mathbf{x}) \in \mathbb{R}^{m \times n}$ denote the Jacobian matrix of $f(\mathbf{x})$, whose elements are:

$$[\mathbf{J}_f(\mathbf{x})]_{i,j} = \frac{\partial}{\partial x_j} f_i(\mathbf{x}).$$

The spectral norm[2] of the Jacobian matrix $\mathbf{J}_f(\mathbf{x})$, denoted by $\|\mathbf{J}_f(\mathbf{x})\|_2$, serves as a lower bound for the Lipschitz constant L Nesterov, 2014; Dherin et al., 2022:

$$\|\mathbf{J}_f(\mathbf{x})\|_2 \leq L, \forall \mathbf{x} \in \mathbb{R}^n. \tag{1}$$

If we consider the relationship between matrix norms, specifically for any matrix $\mathbf{A}$ of rank r over a field of real or complex numbers, we have:

$$\|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F \leq \sqrt{r}\|\mathbf{A}\|_2, \tag{2}$$

where $\|\cdot\|_F$ is the Frobenius norm. This relationship implies that lowering the Frobenius norm below the initial spectral norm will also reduce the spectral norm. The constrained initial gap between these norms ensures that reductions in the Frobenius norm often coincide with decreases in the spectral norm. Such reductions are associated with a lower Lipschitz constant, supported by evidence showing that the Lipschitz constant of neural networks closely aligns with the lower bound defined by (1) (Latorre et al., 2020; Khromov and Singh, 2024).

Definition 2 (L-Lipschitz smoothness)**.** *A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is L-Lipschitz smooth if its gradient is L-Lipschitz continuous:*

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}')\| \leq L\|\mathbf{x} - \mathbf{x}'\|, \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n.$$

An L-Lipschitz continuous function limits how rapidly its output can change with respect to changes in the input. By applying this constraint to the function's gradients, we can establish that they cannot vary sharply and must be bound by a specific value. Put another way, L-Lipschitz smoothness is the upper limit on the curvature of the function, which is inherently linked to the **Hessian** matrix $\mathbf{H}_f(\mathbf{x})$. The Hessian matrix consists of second-order derivatives and is defined for twice-differentiable scalar functions. In the case of a vector-valued function, we can decompose the vector output into scalars where each scalar output has its corresponding Hessian matrix. Generally, the Hessian matrix is a square matrix $\mathbf{H}_f(\mathbf{x}) \in \mathbb{R}^{n \times n}$, whose elements are:

$$[\mathbf{H}_f(\mathbf{x})]_{i,j} = \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}.$$

L-Lipschitz smoothness is equivalent to the condition that the eigenvalues of the Hessian matrix are smaller than L, with the spectral norm as the lower bound of L. Based on (2), we can again use the Frobbenius norm as a proxy, which we can estimate in a computationally efficient manner.

[2]The spectral norm of a matrix $\mathbf{A}$ is defined as $\|\mathbf{A}\|_2 = \max_{\mathbf{x} \neq 0} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$. This norm is equivalent to the largest singular value of $\mathbf{A}$.

3.2 Hutchinson's Estimator

Computing the norms of Jacobian and Hessian matrices is computationally demanding as it requires materializing the whole matrix, which is often infeasible for large neural networks. Hutchinson's estimator (Hutchinson, 1989) offers a practical alternative, allowing the estimation of these norms without needing to compute the entire matrices, thereby achieving feasible computation times. The estimator provides an efficient way to estimate the trace and, consequently, the Frobenius norm of a matrix $\mathbf{A}$, exploiting the relationship $\mathrm{Tr}(\mathbf{A}\mathbf{A}^\top) = \|\mathbf{A}\|_F^2$, where $\mathrm{Tr}(\cdot)$ is the matrix trace. The original formulation uses random normal vectors to estimate the trace of an arbitrary square matrix $\mathbf{B}$:

$$\mathbb{E}[\mathbf{v}^\top \mathbf{B}\mathbf{v}] = \mathrm{Tr}(\mathbf{B}), \tag{3}$$

where $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. With $\mathbf{J}$ as a shorthand for $\mathbf{J}(\mathbf{x})$, we can substitute $\mathbf{B} = \mathbf{J}\mathbf{J}^\top$, which yields:

$$\begin{aligned}\mathbb{E}\left[\mathbf{v}^\top \left(\mathbf{J}\mathbf{J}^\top\right)\mathbf{v}\right] &= \mathrm{Tr}(\mathbf{J}\mathbf{J}^\top) = \|\mathbf{J}\|_F^2 \\ &= \mathbb{E}\left[\left(\mathbf{v}^\top\mathbf{J}\right)\left(\mathbf{v}^\top\mathbf{J}\right)^\top\right] \\ &= \mathbb{E}\left[\left\|\mathbf{v}^\top\mathbf{J}\right\|^2\right],\end{aligned} \tag{4}$$

where the standard 2-norm $\|\cdot\|$ is implied for vectors.[3] By leveraging the relationship in (4), Varga et al. (2018) and Hoffman et al. (2019) employed random projections to compute a Monte Carlo estimate of the Frobenius norm of the Jacobian matrix:

$$\|\mathbf{J}\|_F \approx \sqrt{\frac{1}{p}\sum_{i=1}^{p}\left\|\frac{\partial(\mathbf{v}^{(i)}\mathbf{z})}{\partial\mathbf{x}}\right\|^2}, \tag{5}$$

where $\mathbf{x}$ is the input, $\mathbf{z} = \mathbf{f}(\mathbf{x})$ is the output of a particular network layer, $\mathbf{v}^{(i)}$ is a random vector sampled from the standard normal distribution in the i-th projection, and p is the number of projections. Estimating the norm of the Hessian matrix for the j-th output dimension z_j boils down to:

$$\|\mathbf{H}_j\|_F \approx \sqrt{\frac{1}{p}\sum_{i=1}^{p}\left\|\frac{\partial(\mathbf{v}^{(i)}\frac{\partial z_j}{\partial\mathbf{x}})}{\partial\mathbf{x}}\right\|^2}. \tag{6}$$

[3]We omit the subscript for the vector 2-norm to prevent confusion with the matrix spectral norm.

3.3 JacHess

Our regularization method, JacHess, is designed to minimize the norms of the input-output **Jac**obian and **Hess**ian matrices to promote robustness through Lipschitz continuity and smooth representations. Here, the embedded tokens serve as inputs, and the outputs are the representations of subsequent layers. JacHess adopts Hutchinson's estimator to compute these norms effectively via (5) and (6), maintaining computational efficiency. To circumvent the problem of discrete input space of PLMs, we turn to the **continuous embedding space** by embedding the tokens and then using the embeddings as inputs. Unlike similar methods from the literature, JacHess goes beyond the scope of logits and applies regularization on **intermediate representations** across network layers (i.e., the penultimate layer's outputs).

Dimension Sampling. JacHess considers the high-dimensional nature of intermediate representations. While Mustafa et al. (2020) focus on low-dimensional label spaces constrained by the number of classes—requiring only a few norm estimates—JacHess utilizes the high-dimensional space of intermediate layers. Recognizing that computing the Frobenius norms of Hessian matrices across all dimensions is computationally prohibitive, we integrate a dimension sampling strategy. For a network with K layers, let $D^{(k)}$ denote the random subset of indices for the output dimensions in the k-th layer. We define the regularization term as:

$$\sum_{k=1}^{K}\left(\lambda_1^{(k)}\|\mathbf{J}^{(k)}\|_F + \lambda_2^{(k)}\sum_{d\in D^{(k)}}\|\mathbf{H}_d^{(k)}\|_F\right), \tag{7}$$

where $\lambda_1^{(k)}$ and $\lambda_2^{(k)}$ are the regularization factors. By selecting a random subset of representation dimensions, we reconcile the need for comprehensive regularization with the limits of computational feasibility.

Regularization Dynamics. In addition to applying our method to the training set inputs (JacHess$_{\text{train}}$) by adding the regularization term to the original loss function, we also explore using a separate **unlabeled** dataset (JacHess$_{\text{unlab}}$). In this case, we minimize the regularization term on its own, following a **dual-mode** approach: We first minimize the training loss and then switch to pure

regularization on the unlabeled data, alternating between these two modes.

Regularization Factors. In our empirical analysis, detailed in Section 7, we explore different strategies for selecting the Jacobian and Hessian regularization factors, $\lambda_1^{(k)}$ and $\lambda_2^{(k)}$, from (7). Let $\boldsymbol{\lambda}_i = [\lambda_i^{(1)}, \ldots, \lambda_i^{(K)}]$, $i \in \{1, 2\}$, represent the vectors of Jacobian and Hessian regularization factors across all PLM layers. We experiment with three approaches to applying regularization: uniformly across all layers, in proportion to the smoothness of the base PLM before fine-tuning, and inversely relative to that smoothness. We first compute the norms of the Jacobian matrices for each layer of the PLM, represented as:

$$\mathbf{j} = \left[\|\mathbf{J}^{(1)}\|_F, \|\mathbf{J}^{(2)}\|_F, \ldots, \|\mathbf{J}^{(K)}\|_F \right],$$

where $\mathbf{J}^{(k)}$ is the Jacobian matrix of the k-th layer. We then derive the vector of regularization factors, $\boldsymbol{\lambda}_i$, using $\mathbf{j}$ to achieve inverse proportionality to smoothness, so that smoother layers are assigned lower factors, while using $-\mathbf{j}$ for direct proportionality. Our empirical evaluation in Section 7 demonstrates that scaling the regularization factors proportional to the base PLM smoothness is the most effective among the three approaches. We also tested different methods for scaling, including standard normalization and $softmax$. The results show that $softmax$ slightly outperforms plain normalization, potentially due to its ability to assign greater importance to the smoothest layers, enhancing their influence on the overall performance. We consistently apply the same regularization factors to both the Jacobian and Hessian terms, finding that setting $\boldsymbol{\lambda}_1 = \boldsymbol{\lambda}_2$ (henceforth $\boldsymbol{\lambda}$) is the optimal strategy based on our experiments.

Application to Transformer Models. The JacHess estimators introduced in Section 3.2 were originally formulated for general multilayer neural networks with fixed-dimensional inputs and outputs. In this work, we focus on PLMs based on the transformer architecture (Vaswani et al., 2017), which is more complex than a multilayer network with fixed-dimensional inputs and outputs. While transformers process token embeddings of a fixed dimension (e.g., $\mathbb{R}^d$), the length of the token sequences they handle can vary. To adapt JacHess to transformer-based PLMs, we extend its application to accommodate variable-length token sequences, thereby combining the computational efficiency of JacHess with the architectural complexity of PLMs. In this adaptation, we work directly with continuous token embeddings rather than discrete token identifiers, allowing us to leverage Lipschitz continuity and apply regularization techniques effectively. In particular, we treat each transformer block as a distinct layer, with the embedded tokens serving as inputs and the block's representations as outputs. While the estimators proposed in Section 3.3 focus on the contribution of individual tokens, we capture the cumulative effect by summing the norms over all tokens in a sequence. These norms are computed for each token's representation at each layer and aggregated across both tokens and layers. By summing the layer-wise Frobenius norms of the Jacobian and Hessian matrices, we account for each layer's additive contributions to the network's overall smoothness, ensuring uniform regularization across all layers.

4 Experimental Setup

This section describes the experimental framework, including the models, methods, and datasets used.

4.1 Models

In our empirical analysis, we evaluate the set of decoder-based OPT models (Zhang et al., 2022) at three different scales: 125M, 1.3B, and 6.7B parameters. Additionally, we include the Llama 2 model with 7B parameters (Llama-2-7B; Touvron et al., 2023). We additionally evaluate the encoder-based BERT (Devlin et al., 2019) as a baseline.

4.2 Regularization Methods

We focus on similar representation-based regularization methods, incorporating two of the methods described in the related work. The first is Jacobian regularization (Hoffman et al., 2019), which targets the minimization of the Jacobian norm of the model's logits with respect to the inputs. The second method, Cross-Hölder regularization (Mustafa et al., 2020), builds on Jacobian regularization by additionally aiming to reduce the norm of the input-logit Hessian matrix. We evaluate both of these methods, along with JacHess,

using standard regularization on the training data and the dual-mode approach with separate unlabeled data (cf. Section 3.3), which we refer to as $\texttt{method}_{\textbf{train}}$ and $\texttt{method}_{\textbf{unlab}}$, respectively.

In addition, we include standard L_2 regularization as a baseline. We further compare our method against task-adaptive pre-training (TAPT; Gururangan et al., 2020), which leverages unlabeled data for better generalization, and sharpness-aware minimization (SAM; Foret et al., 2021), where we set the neighborhood size parameter ρ to 0.05. Models trained without any of the mentioned regularization techniques are denoted by BASE.

4.3 Datasets

We evaluate the regularization techniques on the GLUE benchmark (Wang et al., 2018). Aligning with standard practices in assessing in-distribution generalization, we include eight tasks consisting of four binary classification tasks for single sequences (CoLA, SST-2, RTE), three binary classification tasks for sequence pairs (MRPC, QQP, QNLI), one multi-class classification task for sequence pairs (MNLI), and one regression task (STS-B). We evaluate model performance on development sets of the GLUE benchmark. For approaches that leverage unlabeled data, we sample 1000 unlabeled instances from a set not used for testing. When evaluating generalization, we use Matthew's correlation for CoLA, F_1 for MRPC and QQP, Spearman's correlation for STS-B, and accuracy for the remaining datasets. In several experiments, we calculate the average GLUE score across all eight selected tasks, a metric commonly employed in seminal works (Devlin et al., 2019; Houlsby et al., 2019) to capture overarching trends or patterns in model behavior under various regularization strategies.

When evaluating generalization under domain shifts, we use the IMDb sentiment classification dataset (Maas et al., 2011) alongside datasets from the GLUE benchmark. The datasets are paired to match similar tasks across different domains: IMDb is paired with SST-2 for sentiment classification, MRPC and QQP are paired for paraphrase detection, and RTE and QNLI are paired for natural language inference. This setup allows us to assess how well models generalize across domains while performing the same underlying task.

4.4 Fine-Tuning

We fine-tune each language model on GLUE tasks, capping the training set at 10,000 instances for SST-2, MNLI, QNLI, QQP, and IMDb, ensuring computational feasibility for many different experimental setups. With encoder models, we employ the `[CLS]` token's representation, while for decoders, we rely on the representation of the layer's last token. We stack a linear head on top of the final layer and fine-tune the entire model. An exception is made for larger models, namely, OPT-6.7B and Llama 2, where we employ LoRA (Hu et al., 2021) to improve parameter efficiency, setting the rank of decomposition matrices to 8 and α to 16, thus mitigating memory limitations on commodity GPUs.

We run each experiment five times using five different random seeds and report the averages. We use $2 \cdot 10^{-5}$ as the learning rate for standard fine-tuning and $1 \cdot 10^{-4}$ for tuning with LoRA.

5 Enhancing Generalization through Robustness

The goal of our experiments is to evaluate how well JACHESS improves robustness. We begin by testing our method against perturbations in the embedding space, followed by testing its impact on generalization for both in-distribution data and data under domain shift.

5.1 Embedding Perturbation

Our initial experiment aims to assess the robustness of intermediate PLM representations to continuous perturbations in the embedding space, verifying whether JACHESS promotes smooth representations capable of effectively handling changes within that space. We introduce these perturbations by adding noise to the embeddings, specifically by augmenting the token embeddings $\mathbf{e}$ with a noise vector scaled by a factor $\delta \geq 0$. We define the perturbed embedding $\mathbf{e}'$ as:

$$\mathbf{e}' = \mathbf{e} + \delta\mathbf{v},\ \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

where $\mathbf{v}$ is a normally distributed random vector. The degree of perturbation is controlled by δ, providing a quantitative measure of robustness in embedding space.

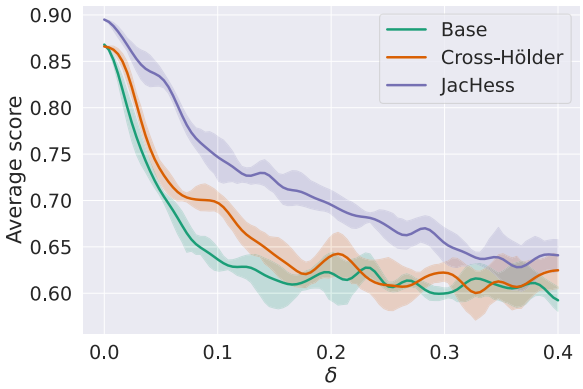


Figure 1: Predictive accuracy with respect to the degree of perturbation δ for Llama 2, averaged across GLUE development datasets. To avoid clutter, we plot the results for the base model without regularization, the Cross-Hölder$_{\text{unlab}}$ method, and JacHess$_{\text{val}}$. Refer to Figure 4 for results concerning other models.

Figure 1 shows the predictive accuracies under embedding perturbation for JacHess together with Cross-Hölder and base. Notably, even as the degree of perturbation increases, JacHess maintains an advantage in predictive accuracy.

5.2 Token Corruption

While the literature elaborates on the link between robustness and generalization in neural networks (Khromov and Singh, 2024), it remains unclear whether and to what extent robustness in the embedding space correlates with robustness in the discrete input space in the case of PLMs. To investigate this, we test how models that are regularized in the continuous embedding space respond to discrete token corruption.

We simulate token corruption by replacing a proportion of tokens with the special ''unknown'' token `[UNK]`. This corruption is applied after fine-tuning at rates of 5%, 10%, 15%, and 20%, and we then evaluate the impact on predictive accuracy. Table 1 shows the results for token corruption. JacHess *maintains the best predictive accuracies across different levels of token corruption.* The most notable differences become more pronounced at more extreme levels of token corruption, specifically at 15% and 20%. JacHess$_{\text{unlab}}$ appears to be more resilient to token corruption than its counterpart JacHess$_{\text{train}}$.

5.3 In-Distribution Generalization

We now focus on evaluating the effect of JacHess on in-distribution generalization. This analysis provides a baseline for understanding how enhancements in robustness influence performance.

		Token corruption [%]			
		5	10	15	20
BERT	base	.736	.721	.703	.671
	L_2	.730	.724	.715	.677
	Jacobian$_{\text{unlab}}$	.744	.729	.710	.680
	Cross-Hölder$_{\text{unlab}}$	.749	.734	.717	.684
	JacHess$_{\text{train}}$	.754	.746	.728	.702
	JacHess$_{\text{unlab}}$	**.759**	**.751**	**.742**	**.729**
OPT-125m	base	.687	.681	.654	.613
	L_2	.694	.685	.672	.639
	Jacobian$_{\text{unlab}}$	.699	.694	.667	.642
	Cross-Hölder$_{\text{unlab}}$	.709	.693	.679	.648
	JacHess$_{\text{train}}$	.719	.710	.702	.689
	JacHess$_{\text{unlab}}$	**.735**	**.731**	**.723**	**.704**
OPT-6.7B	base	.839	.831	.809	.787
	L_2	.846	.830	.805	.792
	Jacobian$_{\text{unlab}}$	.845	.834	.819	.799
	Cross-Hölder$_{\text{unlab}}$	.842	.834	.817	.796
	JacHess$_{\text{train}}$	.847	.836	.821	.804
	JacHess$_{\text{unlab}}$	**.862**	**.851**	**.839**	**.813**
Llama-2-7B	base	.852	.831	.813	.786
	L_2	.857	.835	.816	.793
	Jacobian$_{\text{unlab}}$	.848	.837	.822	.804
	Cross-Hölder$_{\text{unlab}}$	.854	.832	.811	.796
	JacHess$_{\text{train}}$	.860	.849	.827	.808
	JacHess$_{\text{unlab}}$	**.883**	**.869**	**.853**	**.829**

Table 1: Average predictive accuracy with token corruption. We adjust the percentage of token corruption to 10%, 15%, and 20%. For each dataset, we conduct experiments five times using different seeds and report the average score on the GLUE benchmark. Best scores within the same model and token corruption setup are shown in **bold**.

Table 2 shows the predictive accuracies across models, datasets, and regularization methods. We observe that JacHess *consistently outperforms both standard unregularized fine-tuning (*base*) and other regularization baselines*, with absolute improvements in the average GLUE score ranging from 2% to 4.5% compared to unregularized fine-tuning. Moreover, JacHess demonstrates consistent gains as the scale of the models increases. The relative improvement is notably pronounced in the larger models, namely OPT-6.7B and Llama 2. Furthermore, JacHess$_{\text{unlab}}$ generally outperforms JacHess$_{\text{train}}$, a trend also observed for Jacobian and Cross-Hölder methods where using a separate set for regularization proves advantageous over using the training set.

		CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	avg.
BERT	BASE	.466	.894	.852	.855	.810	.700	.832	.610	.752
	L_2	.470	.892	.861	.859	.816	.711	.843	.608	.758
	TAPT	.510	.905	.857	.860	.821	.714	.835	.624	.766
	SAM	.483	.889	.849	.851	.809	.708	.822	.619	.754
	Jacobian$_{\text{train}}$	.471	.883	.850	.847	.806	.704	.821	.607	.749
	Jacobian$_{\text{unlab}}$	.475	.892	.854	.845	.812	.710	.823	.613	.753
	Cross-Hölder$_{\text{train}}$	.498	.901	.842	.839	.824	.707	.830	.605	.756
	Cross-Hölder$_{\text{unlab}}$	.504	.905	.852	.836	**.829**	.712	.838	.616	.762
	JacHess$_{\text{train}}$	.514†	**.912**†	.848	.862	.816	.710	.836	.621†	.765
	JacHess$_{\text{unlab}}$	**.557**†	.906	**.864**†	**.891**†	.828†	**.723**†	**.854**†	**.643**†	**.783**
OPT-125M	BASE	.452	.883	.760	.824	.693	.614	.742	.582	.694
	L_2	.458	.889	.804	.819	.727	.624	.744	.586	.706
	TAPT	.461	.891	.812	.832	**.773**	.653	.759	.596	.722
	SAM	.454	.890	.814	.829	.752	.644	.748	.592	.715
	Jacobian$_{\text{train}}$	.450	.872	.779	.813	.704	.629	.738	.590	.697
	Jacobian$_{\text{unlab}}$	.454	.869	.784	.818	.709	.639	.747	.608	.704
	Cross-Hölder$_{\text{train}}$	.461	.881	.758	.830	.722	.661	.744	.614	.709
	Cross-Hölder$_{\text{unlab}}$	.470	.880	.771	.839	.731	.657	.751	.620	.715
	JacHess$_{\text{train}}$	**.474**†	**.896**†	.819†	.825	.736†	.672†	.757†	.610†	.724
	JacHess$_{\text{unlab}}$	.470†	.884	**.835**†	**.848**†	.768†	**.691**†	**.787**†	**.628**†	**.739**
OPT-1.3B	BASE	.601	.945	.905	.913	.847	.755	.903	.742	.826
	L_2	.596	.948	.910	.907	.872	.785	.911	.739	.834
	TAPT	.612	.941	.908	**.918**	.874	.801	.914	.749	.840
	SAM	.607	.956	.909	.911	.867	.795	.918	.744	.838
	Jacobian$_{\text{train}}$	.589	.940	.911	.908	.851	.770	.892	.731	.824
	Jacobian$_{\text{unlab}}$	.598	.943	.909	.916	.857	.779	.894	.742	.830
	Cross-Hölder$_{\text{train}}$	.612	.939	.910	.902	.879	.773	**.924**	.747	.836
	Cross-Hölder$_{\text{unlab}}$	.608	.949	.909	.907	.884	.771	.921	.750	.837
	JacHess$_{\text{train}}$	.610	.948	.913	.903	.863†	.803†	.918†	.745	.838
	JacHess$_{\text{unlab}}$	**.614**†	**.955**†	**.919**†	.908	**.892**†	**.811**†	.921†	**.751**†	**.846**
OPT-6.7B	BASE	.652	.951	.908	.916	.869	.797	.907	.750	.844
	L_2	.650	.953	.905	.911	.872	.808	.903	.733	.842
	TAPT	.662	.948	.921	.914	.903	.834	**.930**	.754	.858
	SAM	.654	.948	.910	.896	.866	.803	.905	.741	.840
	Jacobian$_{\text{train}}$	.649	.940	.912	.909	.873	.822	.913	.747	.846
	Jacobian$_{\text{unlab}}$	.652	.944	.914	.907	.879	.831	.917	.752	.850
	Cross-Hölder$_{\text{train}}$	.654	.949	.914	.903	.881	.814	.901	.741	.845
	Cross-Hölder$_{\text{unlab}}$	.650	**.959**	.920	.907	.887	.821	.907	.745	.850
	JacHess$_{\text{train}}$	.651	.953	.919†	.911	.889†	.827†	.918†	.750	.852
	JacHess$_{\text{unlab}}$	**.688**†	**.959**	**.928**†	**.922**	**.907**†	**.852**†	.929†	**.776**†	**.870**
Llama-2-7B	BASE	.691	.957	.912	.924	.910	.843	.925	.781	.868
	L_2	.686	.950	.904	.915	.908	.846	.923	.792	.866
	TAPT	.722	.953	.919	.920	.916	.848	.922	.803	.875
	SAM	.682	.961	.914	.920	.911	.846	.925	.794	.869
	Jacobian$_{\text{train}}$	.681	.940	.893	.903	.882	.837	.912	.764	.852
	Jacobian$_{\text{unlab}}$	.693	.955	.915	.913	.890	.844	.923	.769	.863
	Cross-Hölder$_{\text{train}}$	.688	.951	.909	.915	.914	.832	.927	.759	.862
	Cross-Hölder$_{\text{unlab}}$	.691	.949	.913	.917	.909	.838	.931	.779	.866
	JacHess$_{\text{train}}$	.712†	.962	.908	.921	.919†	.851†	.933	.798†	.876
	JacHess$_{\text{unlab}}$	**.746**†	**.973**†	**.951**†	**.934**†	**.929**†	**.872**†	**.940**†	**.813**†	**.895**

Table 2: In-distribution generalization scores on the GLUE development sets averaged across five different seeds for each dataset. The last column shows the average score across datasets. The highest score for each dataset is in **bold**, while the second-highest scores are underlined. The symbol ‘‘†’’ indicates significant differences between JacHess variants and the BASE model, as determined by two-sided Mann-Whitney U tests with $p < .05$, corrected for multiple comparisons using the Holm-Bonferroni method.

		IMDb→SST-2	SST-2→IMDb	RTE→QNLI	QNLI→RTE	MRPC→QQP	QQP→MRPC
OPT-6.7B	BASE	.832	.789	.742	.794	.693	.764
	Jacobian$_{\text{unlab}}$	.865	.798	.772	.804	.707	.772
	Cross-Hölder$_{\text{unlab}}$	.853	**.809**	.779	.809	.713	.776
	JacHess$_{\text{unlab}}$	**.879**	.804	**.795**	**.815**	**.729**	**.810**
Llama-2	BASE	.892	.824	.768	.832	.741	.790
	Jacobian$_{\text{unlab}}$	.904	.832	.776	.842	.761	.799
	Cross-Hölder$_{\text{unlab}}$	.901	**.846**	.780	.848	.754	.812
	JacHess$_{\text{unlab}}$	**.915**	.838	**.796**	**.852**	**.787**	**.861**

Table 3: Generalization scores for target datasets under domain shifts for different ''source → target'' dataset pairs. The highest score for each pair is shown in **bold**.

5.4 Generalization Under Domain Shift

To evaluate how well JacHess handles domain-shift scenarios, we conduct experiments where we simulate these shifts by pairing the datasets that represent the same task but are drawn from different domains. Table 3 shows the generalization scores under domain shifts. We fine-tune the model using unlabeled data only from the source dataset and then evaluate the model on the target dataset. JacHess demonstrates improvements in generalization under domain shifts. In particular, JacHess$_{\text{unlab}}$ consistently outperforms the BASE models across all dataset pairs, showing especially notable gains in paraphrase detection, with absolute increases of 4.6% and 7.1% for MRPC → QQP and QQP → MRPC, respectively. Moreover, JacHess$_{\text{unlab}}$ outperforms other methods on five out of six dataset pairs (all but SST-2 → IMDb) for both OPT-6.7B and Llama 2.

The performance improvements suggest that the smoothness constraints imposed by JacHess help maintain robustness even under domain shifts. These improvements align with the gains observed for in-distribution generalization, underscoring the broader applicability and robustness of JacHess.

6 Calibration

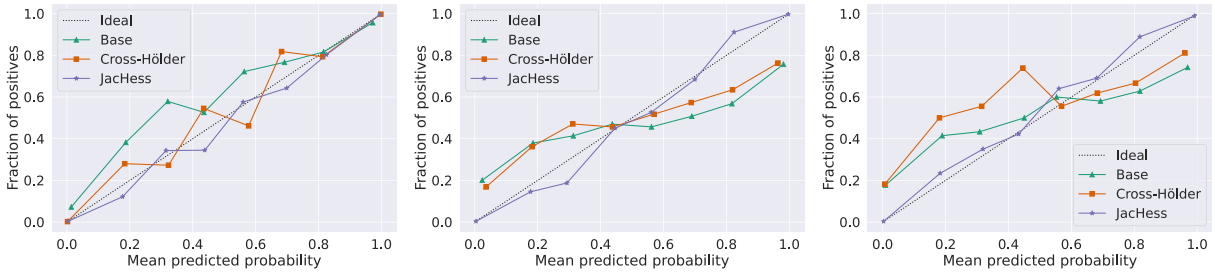
Uncertainty quantification in NLP models has garnered significant attention lately, given its importance in various applications (Abdar et al., 2021; Xiao and Wang, 2019; Xiao et al., 2022). Motivated by research indicating a connection between smoothness in model representations and more reliable uncertainty quantification (Lakshminarayanan et al., 2017; Van Amersfoort et al., 2020), we explore the effect of JacHess on out-of-the-box calibration, using the PLM's softmax outputs as confidence scores (Desai and Durrett, 2020). We utilize the Brier score, which quantifies uncertainty by measuring the mean squared distance between predicted probabilities and actual outcomes.[4] A lower Brier score indicates better calibration. In addition to Brier scores, we report the expected calibration errors (ECEs) and provide corresponding calibration plots.

	BASE	Cross-Hölder	JacHess
Model	*Brier/ECE*	*Brier/ECE*	*Brier/ECE*
BERT	.213/.042	.202/.039	**.184**/**.035**
OPT-125M	.256/.057	.233/.045	**.204**/**.040**
OPT-1.3B	.184/.038	.193/.042	**.157**/**.029**
OPT-6.7B	.189/.037	.157/.032	**.094**/**.024**
Llama-2-7B	.167/.034	.163/.031	**.089**/**.017**

Table 4: Average Brier scores and ECE on the development sets across binary classification tasks in the GLUE benchmark (CoLA, SST-2, MRPC, RTE, QQP, and QNLI). Regularization methods are applied on a separate, unlabeled set. Lower Brier scores and ECE indicate better performance. ECE was calculated using eight bins to ensure detailed calibration analysis. Best scores are highlighted in **bold**.

Table 4 shows the Brier scores for different models and regularization methods. Besides enhancing generalization, JacHess also contributes to better uncertainty estimation, manifested in a lower Brier score. We observe that JacHess *surpasses baseline methods, offering improved calibration with lower Brier scores and ECEs, underscoring its effectiveness in producing reliable uncertainty estimates*. Further illustrating

[4]The Brier score can be decomposed into two components: calibration, which reflects how well predicted probabilities align with observed frequencies, and refinement, which measures how spread out the confidence scores are (DeGroot and Fienberg, 1983).



(a) CoLA (b) MRPC (c) QQP

Figure 2: Calibration plots for six binary classification datasets from the GLUE benchmark for the Llama 2 model. Results are accumulated in five seeds with eight bins for calculating mean predicted probability. Refer to Figure 5 in the Appendix for the results of other binary classification tasks.

these enhancements, Figure 2 shows the calibration plots for binary classification tasks from the GLUE benchmark. Here, JacHess closely aligns the PLM's calibration curve with the ideal curve, outperforming both Cross-Hölder regularization and BASE.

7 Analysis

Several design choices behind JacHess may impact its efficacy. We next analyze these factors.

7.1 Scope and Degree of Regularization

We begin by analyzing how the **scope** of regularization affects model performance—specifically, by applying JacHess to the logits versus across multiple network layers. Additionally, we evaluate the impact of the **degree** of regularization on different layers by controlling the regularization factors $\boldsymbol{\lambda}$. Table 5 compares these design choices, previously laid out in Section 3.3. *We observe that initializing the regularization factors in proportion to the base PLM smoothness across layers proves to be the most effective approach.* Applying uniform regularization across all layers surpasses the scope of only regularizing the logits. In contrast, initializing the regularization coefficients to be inversely proportional to the base smoothness for each layer detrimentally affects performance.

7.2 Dimension Sampling

We investigate the impact on model performance by regularizing varying numbers of Frobenius norms of Hessian matrices, each corresponding to specific dimensions of a layer's representation. Due to the computational intensity of estimating the norms for Hessian matrices, we sample up to 50 different dimensions for regularization.

	Strategy				
Model	Logits	Uni	Inv	Norm	Soft
BERT	.743	.775	.733	.778	**.783**
OPT-125M	.709	.726	.692	.735	**.739**
OPT-1.3B	.832	.845	.803	**.848**	.846
OPT-6.7B	.851	.868	.811	.865	**.870**
Llama-2-7B	.874	.883	.819	.892	**.895**

Table 5: Comparison of strategies for $\text{JacHess}_{\text{unlab}}$ regularization at different application points evaluated on the GLUE development sets. Average predictive accuracy for each model is reported across datasets, based on five runs per dataset using different seeds. **Log** regularizes only the norms corresponding to the Hessian matrices of logits of the penultimate layer. **Uni** applies regularization uniformly across all layers. **Inv** sets $\boldsymbol{\lambda}$ inversely proportional to the base model's smoothness, and **Norm** aligns $\boldsymbol{\lambda}$ directly with the base PLM's smoothness, while **Soft** applies softmax instead of standard normalization. The highest score for each model is shown in **bold**.

Table 6 shows the impact of adjusting the number of dimensions involved in the Hessian part of JacHess regularization on model performance. Predictive accuracy peaks at 10 dimensions, with diminishing returns as dimensions increase. Expanding to 20 dimensions offers no further improvement, while using 50 dimensions degrades performance, likely due to excessive smoothing.

7.3 Number of Unlabeled Instances

Lastly, we examine the impact of varying the number of unlabeled instances used by $\text{JacHess}_{\text{unlab}}$ on model performance. Figure 3 shows the average scores on the GLUE datasets for both Llama 2 and OPT-6.7B models as the amount of unlabeled data

Model	Number of sampled dimensions				
	0	5	10	20	50
BERT	.764	.771	**.783**	.781	.754
OPT-125M	.698	.721	**.739**	.737	.713
OPT-1.3B	.821	.839	.846	**.850**	.831
OPT-6.7B	.849	.854	.868	**.870**	.857
Llama-2-7B	.873	.886	**.895**	.892	.880

Table 6: Comparison of predictive accuracies of JacHess$_{\text{unlab}}$ using different numbers of dimension for the Hessian part of regularization. Each number corresponds to the dimensions sampled from a specific layer's output. Average scores for the GLUE benchmark development sets are reported based on five runs per dataset. The highest scores for each model are highlighted in **bold**.

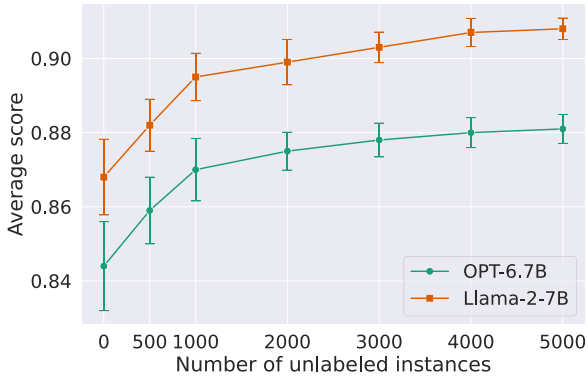


Figure 3: Average scores on the GLUE datasets for Llama 2 and OPT-6.7B models with respect to the number of unlabeled instances used in JacHess$_{\text{unlab}}$. Error bars represent standard deviations over five runs.

increases. Without any unlabeled data, the setup corresponds to the BASE model. Using 500 unlabeled instances for the JacHess estimator already yields a large improvement in generalization as well as a slight reduction in standard deviation. The generalization scores increase sharply when the number of unlabeled examples rises from 500 to 1000, with only marginal gains beyond this point. However, the standard deviation continues to decrease as more unlabeled data is added, reflecting greater stability across runs. This pattern suggests that incorporating more than 1000 unlabeled instances contributes significantly to the robustness of PLMs on the GLUE datasets, though the primary benefits are realized within the initial increase to 1000 examples.

While including more unlabeled instances can enhance performance and stability, it also affects the runtime of JacHess. Based on runtimes averaged over 10 runs of Llama 2 on the GLUE datasets, the dual mode approach increases training time by an average factor of 3.2 when the unlabeled set matches the training set size.[5] In our experiments in Sections 5 and 6, we used 1000 unlabeled instances by default, which is typically one-tenth the size of the training set of the GLUE datasets. Since the runtime scales linearly with the number of unlabeled instances, this results in a more manageable runtime increase of approximately $1 + \frac{3.2-1}{10} = 1.22$ times, making it feasible for real-world applications.

8 Conclusion

The success of PLMs depends significantly on their robustness and ability to generalize. In this work, we examined how enforcing representation smoothness affects these properties in transformer-based PLMs by leveraging Jacobian and Hessian norm minimization. We introduced JacHess, a regularization method that enforces smooth representations with respect to the input embeddings, which serve as an alternative to discrete tokens. Leveraging computationally efficient estimators for the Jacobian and Hessian norms, JacHess is practical and effective, consistently surpassing unregularized fine-tuning and similar regularization approaches by improving both in-distribution and cross-domain generalization. Across diverse PLMs and GLUE benchmark tasks, JacHess enhances predictive accuracy and model calibration, leading to more reliable uncertainty quantification. These results highlight the importance of representation smoothness for model robustness, setting JacHess as the new reference for improving the generalization and calibration of PLMs.

Acknowledgments

We sincerely thank Dani Yogatama, our TACL action editor, and the anonymous reviewers for their insightful feedback and constructive suggestions.

References

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad

[5]Runtime estimates are based on 1000 instances each for the training and unlabeled sets.

Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, et al. 2021. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297. https://doi.org/10.1016/j.inffus.2021.05.008

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2021. Better fine-tuning by reducing representational collapse. In *International Conference on Learning Representations*.

Dara Bahri, Hossein Mobahi, and Yi Tay. 2022. Sharpness-aware minimization improves language model generalization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7360–7371, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.508

Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. 2017. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Sébastien Bubeck and Mark Sellke. 2021. A universal law of robustness via isoperimetry. *Advances in Neural Information Processing Systems*, 34:28811–28822.

Wojciech M. Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. 2017. Sobolev training for neural networks. *Advances in Neural Information Processing Systems*, 30.

Morris H. DeGroot and Stephen E. Fienberg. 1983. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 32(1–2):12–22. https://doi.org/10.2307/2987588

Zhun Deng, Hangfeng He, and Weijie Su. 2021. Toward better generalization bounds with locally elastic stability. In *International Conference on Machine Learning*, pages 2590–2600. PMLR.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.21

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

Benoit Dherin, Michael Munn, Mihaela Rosca, and David Barrett. 2022. Why neural networks find simple solutions: The many regularizers of geometric complexity. *Advances in Neural Information Processing Systems*, 35:2333–2349.

Harris Drucker and Yann Le Cun. 1992. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6):991–997. https://doi.org/10.1109/72.165600, PubMed: 18276495

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2021. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144. https://doi.org/10.1145/3422622

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J. Cree. 2021. Regularisation of neural networks by enforcing Lipschitz continuity. *Machine Learning*, 110:393–416. https://doi.org/10.1007/s10994-020-05929-w

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't

stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.740

Judy Hoffman, Daniel A. Roberts, and Sho Yaida. 2019. Robust learning with Jacobian regularization. *arXiv preprint arXiv:1908.02729v1*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J. Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Dieuwke Hupkes, Mario Giulianelli, Verna Dankers, Mikel Artetxe, Yanai Elazar, Tiago Pimentel, Christos Christodoulopoulos, Karim Lasri, Naomi Saphra, Arabella Sinclair, Dennis Ulmer, Florian Schottmann, Khuyagbaatar Batsuren, Kaiser Sun, Koustuv Sinha, Leila Khalatbari, Maria Ryskina, Rita Frieske, Ryan Cotterell, and Zhijing Jin. 2023. A taxonomy and review of generalization research in NLP. *Nature Machine Intelligence*, 5(10):1161–1174. https://doi.org/10.1038/s42256-023-00729-y

Michael F. Hutchinson. 1989. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076. https://doi.org/10.1080/03610918908812806

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.197

Kenji Kawaguchi, Zhun Deng, Kyle Luh, and Jiaoyang Huang. 2022. Robustness implies generalization via data-dependent generalization bounds. In *International Conference on Machine Learning*, pages 10866–10894. PMLR.

Grigory Khromov and Sidak Pal Singh. 2024. Some intriguing aspects about Lipschitz continuity of neural networks. In *International Conference on Learning Representations*.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, 30.

Fabian Latorre, Paul Rolland, and Volkan Cevher. 2020. Lipschitz constant estimation of neural networks via sparse polynomial optimization. In *International Conference on Learning Representations*.

Binghui Li, Jikai Jin, Han Zhong, John Hopcroft, and Liwei Wang. 2022. Why robust generalization in deep learning is difficult: Perspective of expressive power. *Advances in Neural Information Processing Systems*, 35:4370–4384.

Guangliang Liu, Zhiyu Xue, Xitong Zhang, Kristen Johnson, and Rongrong Wang. 2023. PAC-tuning: Fine-tuning pre-trained language models with PAC-driven perturbed gradient descent. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12178–12189, Singapore. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.748

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

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

Oregon, USA. Association for Computational Linguistics.

Waleed Mustafa, Robert A. Vandermeulen, and Marius Kloft. 2020. Input Hessian regularization of neural networks. In *Workshop on ''Beyond first-order methods in ML systems'' at the 37th International Conference on Machine Learning*.

Yurii Nesterov. 2014. *Introductory Lectures on Convex Optimization: A Basic Course*, 1 edition. Springer Publishing Company, Incorporated.

Itsuki Okimura, Machel Reid, Makoto Kawano, and Yutaka Matsuo. 2022. On the impact of data augmentation on downstream performance in natural language processing. In *Proceedings of the Third Workshop on Insights from Negative Results in NLP*, pages 88–93, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.insights-1.12

Alexander G. Ororbia II, Daniel Kifer, and C. Lee Giles. 2017. Unifying adversarial training algorithms with data gradient regularization. *Neural Computation*, 29(4):867–887. https://doi.org/10.1162/NECO_a_00928, PubMed: 28095194

Mihaela Rosca, Theophane Weber, Arthur Gretton, and Shakir Mohamed. 2020. A case for new neural network smoothness constraints. In *Proceedings on ''I Can't Believe It's Not Better!'' at NeurIPS Workshops*, volume 137 of *Proceedings of Machine Learning Research*, pages 21–32. PMLR.

Walter Rudin. 1964. *Principles of Mathematical Analysis*, volume 3. McGraw-Hill New York.

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. 2018. Adversarially robust generalization requires more data. *Advances in Neural Information Processing Systems*, 31.

Tom Sherborne, Naomi Saphra, Pradeep Dasigi, and Hao Peng. 2024. TRAM: Bridging trust regions and sharpness aware minimization. In *International Conference on Learning Representations*.

Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel R. D. Rodrigues. 2017. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280. https://doi.org/10.1109/TSP.2017.2708039

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288v2*.

Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. 2020. Uncertainty estimation using a single deep deterministic neural network. In *International Conference on Machine Learning*, pages 9690–9700. PMLR.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. New York. Springer. https://doi.org/10.1007/978-1-4757-2440-0

Dániel Varga, Adrián Csiszárik, and Zsolt Zombori. 2018. Gradient regularization improves accuracy of discriminative models. *Schedae Informaticae*, 27:31–45. https://doi.org/10.4467/20838476SI.18.003.10408

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Aladin Virmaux and Kevin Scaman. 2018. Lipschitz regularity of deep neural networks: Analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/W18-5446

Ruigang Wang and Ian Manchester. 2023. Direct parameterization of Lipschitz-bounded deep networks. In *International Conference on Machine Learning*, pages 36093–36110. PMLR.

Xing Wu, Chaochen Gao, Meng Lin, Liangjun Zang, and Songlin Hu. 2022. Text smoothing: Enhance various data augmentation methods on text classification tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 871–875, Dublin, Ireland. Association for Computational Linguistics.

Yuxin Xiao, Paul Pu Liang, Umang Bhatt, Willie Neiswanger, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2022. Uncertainty quantification with pre-trained language models: A large-scale empirical analysis. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7273–7284. https://doi.org/10.18653/v1/2022.findings-emnlp.538, PubMed: 35674204

Yijun Xiao and William Yang Wang. 2019. Quantifying uncertainties in natural language processing tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7322–7329. https://doi.org/10.1609/aaai.v33i01.33017322

Huan Xu and Shie Mannor. 2012. Robustness and generalization. *Machine Learning*, 86:391–423. https://doi.org/10.1007/s10994-011-5268-1

Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. 2021. Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1063–1077. https://doi.org/10.18653/v1/2021.naacl-main.84

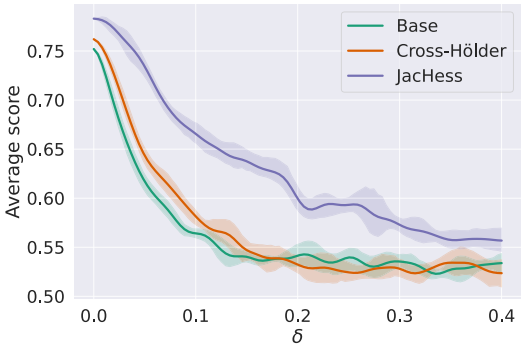
Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2021. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115. https://doi.org/10.1145/3446776

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068v4*.

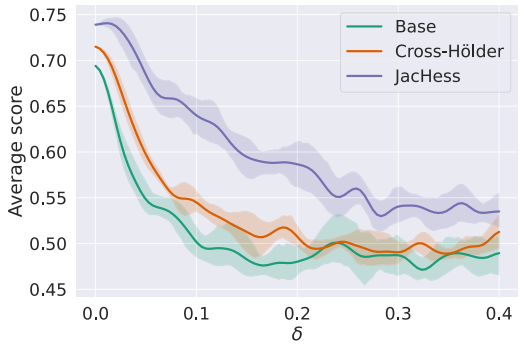
Jing Zhou, Yanan Zheng, Jie Tang, Li Jian, and Zhilin Yang. 2022. FlipDA: Effective and robust data augmentation for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8646–8665, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.592

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. FreeLB: Enhanced adversarial training for natural language understanding. In *International Conference on Learning Representations*.

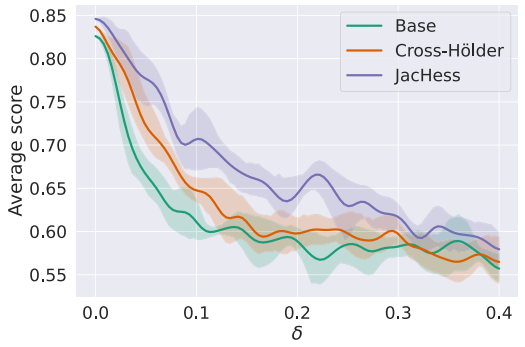
A Complementary Results



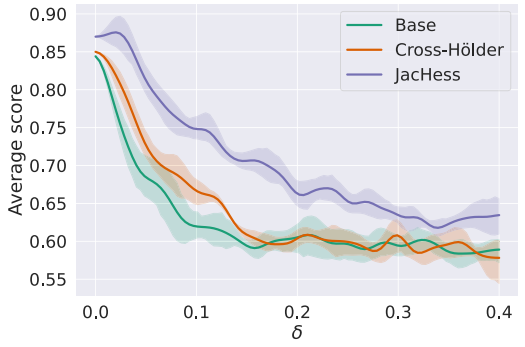
(a) BERT



(b) OPT-125M

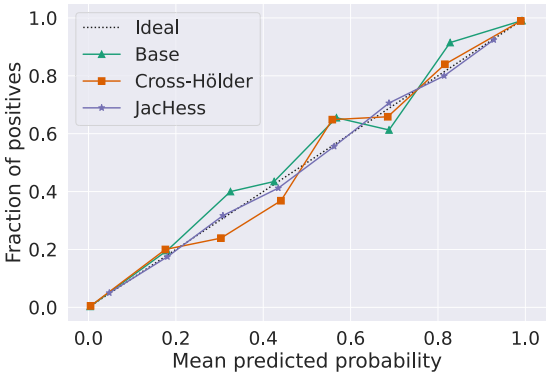


(c) OPT-1.3B

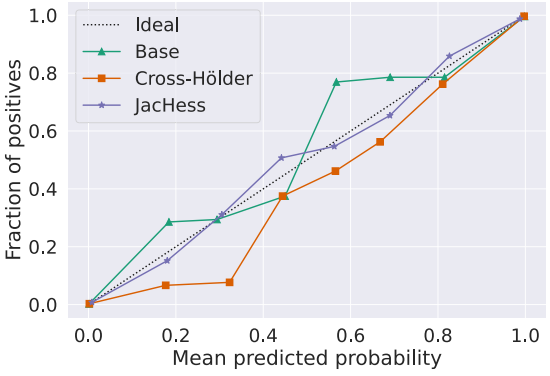


(d) OPT-6.7B

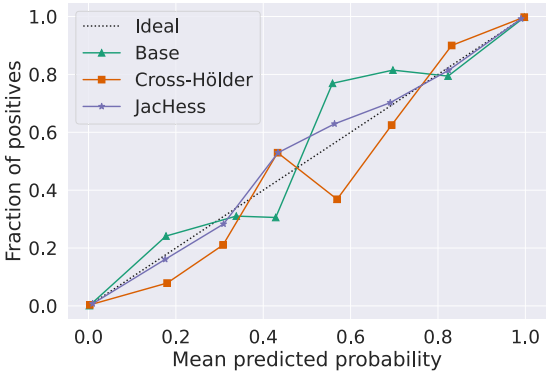
Figure 4: Perturbation in the embedding space. Complementary to Figure 1.



(a) SST-2



(b) RTE



(c) QNLI

Figure 5: Calibration plots for binary classification datasets for LLaMA-2. Complementary to Figure 2.