TDK-thesis

David Leonard Nagy

VectorShake: Few-Shot Text Classification via Latent Space Regularization and Dynamic Adaptive Curriculum

EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

DEPT. OF ARTIFICIAL INTELLIGENCE

Author:

David Leonard Nagy

High School Student

XII. grade

Supervisor:

Gyöngyössy Natabara Máté

PhD Student

Budapest, 2025

Contents

Chapter 1

Introduction

The successful application of large pre-trained language models, such as BERT [1], across a wide range of Natural Language Processing (NLP) tasks often relies on the availability of substantial labeled datasets for fine-tuning. However, in many real-world scenarios and specialized domains, acquiring such large datasets is prohibitively expensive or impractical. This motivates the field of few-shot learning, which aims to develop models capable of generalizing effectively from only a handful of labeled examples per class. While standard fine-tuning can struggle with overfitting or poor generalization under these data-scarce conditions, specialized few-shot methods have emerged. Techniques like SetFit [2] have shown strong performance but can introduce significant computational overhead during the training phase. Therefore, the challenge remains to develop methods that achieve both high accuracy and computational efficiency for few-shot text classification using pre-trained transformers.

To address this challenge, we propose VectorShake, a novel training framework designed to enhance the robustness and few-shot generalization capabilities of BERT-based models. The core idea behind VectorShake is to improve the quality and stability of the model's learned representations by applying a suite of regularization techniques directly within the model's latent space, rather than solely relying on input-level augmentations or complex meta-learning strategies. Specifically, VectorShake integrates five distinct latent-space regularization objectives: representation reconstruction, adversarial perturbation using the Fast Gradient Sign Method (FGSM), contrastive separation between clean and adversarial representations, consistency regularization via Kullback-Leibler (KL) divergence between clean and perturbed predictions, and latent-space mixup also enforced with a KL divergence loss. We hypothesize that applying this combination of regulariz-

ers directly to the internal embeddings encourages the model to learn more robust and transferable features, beneficial when training data is limited.

Furthermore, managing the interplay and intensity of these regularization techniques introduces significant hyperparameter complexity. To mitigate this and potentially enhance performance further, VectorShake incorporates a second key innovation: a Dynamic Adaptive Curriculum (DAC). The DAC mechanism automatically adjusts critical regularization hyperparameters during training – specifically, the FGSM perturbation magnitude (ε), the mixup interpolation parameters (β_a, β_b), and the weight of the mixup loss (λ_{mix}) – based on the Exponential Moving Average (EMA) of the training accuracy and mixup loss. This allows the training process to dynamically adapt the regularization strategy based on ongoing performance feedback, potentially finding better optima and reducing sensitivity to initial hyperparameter settings.

This paper makes the following contributions: First, we propose the VectorShake framework, detailing its combination of multiple latent-space regularization techniques and the novel Dynamic Adaptive Curriculum. Second, through comprehensive ablation studies on the AG News and TREC-6 datasets, we demonstrate the significant and consistent performance benefit derived from the DAC, alongside validating the positive contributions of the individual latent regularization components. Third, we evaluate VectorShake on few-shot benchmarks derived from AG News, GoEmotions, and TREC-6, showing it achieves competitive accuracy, notably exceeding baselines on AG News, while offering substantial training time reductions compared to the computationally intensive SetFit method. Fourth, we analyze the results and ablation studies to provide insights into the effectiveness of different components and potential dataset-dependent interactions.

The remainder of this paper details the VectorShake methodology in Chapter 3, outlines the experimental setup including datasets, baselines, and implementation details in Chapter 4, presents the main comparative results in Chapter 5, analyzes the ablation studies in Chapter 6, discusses the broader implications and limitations in Chapter 7, and finally concludes in Chapter 8.

Chapter 2

Related Work

Our work on VectorShake intersects with several active research areas in natural language processing, primarily few-shot text classification, latent space regularization techniques, and adaptive training methods.

2.1 Few-Shot Text Classification

Learning effectively from limited labeled data is a critical challenge in NLP. Various paradigms have been explored to address few-shot text classification. Meta-learning approaches, including optimization-based methods like MAML [3] and metric-based methods like Prototypical Networks [4] and Relation Networks [5], aim to learn adaptable models or embedding spaces that generalize quickly to new tasks with few examples. Data augmentation techniques, such as back-translation or using generative models, attempt to synthetically increase the amount of available training data [6]. More recently, leveraging the power of large pre-trained language models (PLMs) like BERT [1] has become central. Strategies include prompt-based learning [7, 8], which reformulates downstream tasks as language modeling problems, and advanced fine-tuning techniques. SetFit [2] represents a prominent example of the latter, employing contrastive learning on sentence pairs to fine-tune sentence transformers efficiently for few-shot scenarios. Our work uses SetFit as a key baseline due to its strong performance and relevance. Unlike prompt-based methods or SetFit's contrastive sentence-pair approach, VectorShake focuses on enhancing the robustness of the underlying PLM's representations during fine-tuning through a combination of targeted regularizations applied directly within the model's latent space.

2.2 Latent Space Regularization Techniques in NLP

Manipulating or regularizing the internal latent representations of neural networks has been explored to improve model robustness and generalization. Adversarial training, commonly applied at the input level [9, 10], has also been investigated in the embedding or latent space [11, 12] to enhance robustness against small perturbations in representation. Consistency regularization encourages the model to produce similar outputs for perturbed versions of an input, often using KL divergence or mean squared error between predictions on clean and augmented samples [13, 14]. While typically applied to input augmentations, VectorShake applies consistency objectives between clean and adversarially perturbed latent representations. Mixup [15], which linearly interpolates input examples and their labels, has shown benefits for generalization. Extensions have applied mixup to hidden states or embeddings [16]. VectorShake adopts this idea but performs mixup specifically between adversarial latent representations and enforces consistency with clean predictions via KL divergence. Reconstruction losses, common in autoencoders, are sometimes used as auxiliary tasks during supervised training to encourage more informative representations [17]. Contrastive learning aims to pull representations of similar items together while pushing dissimilar ones apart [18, 19] and is employed by SetFit at the sentence level. VectorShake utilizes a contrastive objective within its latent space specifically between clean and adversarial representations derived from the same input. While these individual regularization techniques have been explored, VectorShake's novelty lies in the synergistic combination of multiple distinct latent-space strategies — reconstruction, adversarial perturbation, contrastive separation, consistency enforcement, and latent mixup — within a unified framework targeted specifically at improving few-shot text classification.

2.3 Curriculum Learning and Adaptive Methods

Curriculum learning traditionally involves presenting training examples in a meaningful order, typically from easy to hard, to improve convergence and generalization [20]. While effective for certain tasks, this usually involves predefined data ordering or task scheduling. Adaptive methods, which adjust aspects of the training process dynamically, offer another avenue. Learning rate schedulers or adaptive optimizers like Adam [21] are ubiquitous. Some work has explored adapting other hyperparameters, such as dropout

rates or regularization weights, during training, although this is less common than standard curriculum learning or adaptive optimization. The Dynamic Adaptive Curriculum (DAC) proposed in VectorShake differs from these approaches. It does not reorder data but instead dynamically modulates the strength and nature of the latent regularization itself (specifically, FGSM epsilon, mixup weighting, and mixup distribution parameters) based on ongoing performance feedback (EMA of training accuracy and a relevant loss component). We use the training accuracy for this adaptation, as using validation metrics would require either a larger K or would result in data contamination. This represents an adaptive regularization strategy designed to optimize the complex interplay between multiple regularization techniques during training, potentially reducing sensitivity to initial hyperparameter choices and finding better training trajectories, especially in challenging low-data settings.

Chapter 3

Methodology: VectorShake

The VectorShake framework aims to improve few-shot text classification by enhancing the robustness and generalization of a base pre-trained transformer model through a combination of latent space regularization techniques coupled with a Dynamic Adaptive Curriculum (DAC). We build upon a standard BERT architecture and introduce several auxiliary objectives and an adaptive mechanism during the fine-tuning process. Our code is available on GitHub.

3.1 Base Architecture and Latent Representation

We utilize a standard BERT model [1] as our base encoder. Given an input text sequence x, the BERT encoder E with parameters θ_e produces contextualized hidden states. We extract the latent representation h from the final hidden state corresponding to the special [CLS] token:

$$h = E(x; \theta_e)_{[CLS]} \tag{3.1}$$

where $h \in \mathbb{R}^d$ and d is the hidden dimension size of the BERT model. This latent representation h encapsulates semantic information from the input text. A linear classification layer C with parameters θ_c is applied on top of h to produce logits z for the target classes:

$$z = C(h; \theta_c) \tag{3.2}$$

The standard training objective for classification is the Cross-Entropy loss $\mathscr{L}_{CE}$ between the predicted probabilities (obtained via softmax(z)) and the true one-hot label y:

$$\mathscr{L}_{CE} = -\sum_i y_i \log(\text{softmax}(z)_i) \tag{3.3}$$

3.2 Latent Space Regularization Techniques

VectorShake augments the standard $\mathscr{L}_{CE}$ with five regularization terms operating primarily on the latent representation h. Our implementation, referred to as 'EnhancedBert', incorporates these mechanisms.

3.2.1 Latent Reconstruction ($\mathscr{L}_{rec}$)

To encourage h to retain comprehensive information, we introduce an auxiliary reconstruction task. A linear layer, Reconstruct, parameterized by θ_{rec}, attempts to reconstruct h from itself. The reconstruction loss $\mathscr{L}_{rec}$ is the Mean Squared Error (MSE) between the original and reconstructed vectors:

$$h_{rec} = \text{Reconstruct}(h; \theta_{rec}) \tag{3.4}$$

$$\mathscr{L}_{rec} = \text{MSE}(h, h_{rec}) = ||h - h_{rec}||_2^2 \tag{3.5}$$

This acts as a regularizer, penalizing representations that lose information.

3.2.2 Latent Adversarial Perturbation

To improve robustness, we employ FGSM [9] on the latent vector h. We compute the gradient of $\mathscr{L}_{CE}$ with respect to h and create an adversarial latent vector h_{adv} by moving h slightly in the direction of the gradient's sign, scaled by ε:

$$\nabla_h \mathscr{L}_{CE}(h, y) = \frac{\partial \mathscr{L}_{CE}(C(\text{detach}(h); \theta_c), y)}{\partial h} \tag{3.6}$$

$$h_{adv} = h + \varepsilon \cdot \text{sign}(\nabla_h \mathscr{L}_{CE}(h, y)) \tag{3.7}$$

We generate two such samples, h_{adv1} and h_{adv2}, per input.

3.2.3 Latent Contrastive Loss ($\mathscr{L}_{con}$)

This loss encourages distinguishing h from h_{adv1} and h_{adv2}, maximizing their distance (minimizing similarity). Using cosine similarity $\text{sim}(u, v) = \frac{u \cdot v}{||u|| ||v||}$;

$$\mathscr{L}_{con} = \mathbb{E}[2 - \text{sim}(h, h_{adv1}) - \text{sim}(h, h_{adv2})] \tag{3.8}$$

Minimizing $\mathscr{L}_{con}$ pushes adversarial representations away from the original.

3.2.4 Latent Mixup and Consistency Loss ($\mathscr{L}_{mix}$)

Inspired by Mixup [15], we interpolate between adversarial examples in latent space. For a batch of h_{adv1} we create a shuffled version $h_{adv1,shuffled}$. We sample $\alpha \sim \text{Beta}(\beta_a, \beta_b)$ and compute the mixed representation h_{mix}:

$$h_{mix} = \alpha \cdot h_{adv1} + (1 - \alpha) \cdot h_{adv1,shuffled} \tag{3.9}$$

We enforce prediction consistency between h and h_{mix} using KL divergence. Let $p_{clean} = \text{softmax}(C(h; \theta_c))$ and $p_{mix} = \text{softmax}(C(h_{mix}; \theta_c))$;

$$\mathscr{L}_{mix} = D_{KL}(\text{detach}(p_{clean}) || p_{mix}) \tag{3.10}$$

This encourages consistent predictions for interpolations between adversarial points. We use the numerically stable $log\_softmax$ input for implementation.

3.2.5 Latent Adversarial Consistency Loss ($\mathscr{L}_{cons}$)

Similarly, we enforce consistency between the original h and a direct adversarial perturbation h_{adv1} using KL divergence:

$$\mathscr{L}_{cons} = D_{KL}(\text{detach}(p_{clean}) || p_{adv1}) \tag{3.11}$$

where $p_{adv1} = \text{softmax}(C(h_{adv1}; \theta_c))$. This encourages robustness to direct FGSM perturbations.

3.3 Dynamic Adaptive Curriculum (DAC)

Managing the hyperparameters (ε, β_a, β_b, λ_{mix} etc.) is challenging. VectorShake uses a DAC to adjust ε, β_a, β_b and λ_{mix} dynamically. The DAC monitors the EMA of the training accuracy (as using the validation accuracy for these purposes would result in data contamination) (EMA_{acc}) and mixup loss (EMA_{mix}). Let $EMA_{train}(t)$ at epoch t be computed with smoothing factor α_{ema}:

$$EMA_{val}(t) = \alpha_{ema} \cdot \text{current\_value}(t) + (1 - \alpha_{ema}) \cdot EMA_{train}(t - 1) \tag{3.12}$$

Based on changes $\Delta EMA_{acc} = EMA_{acc}(t) - EMA_{acc}(t-1)$ and $\Delta EMA_{mix} = EMA_{mix}(t-1) - EMA_{mix}(t)$ relative to thresholds (τ_{acc}, τ_{mix}) and update strength (η_{DAC}), the DAC

updates the parameters. The logic implemented in the dynamic_curriculum_ema function generally follows these principles:

- If $\Delta EMA_{acc} > \tau_{acc}$: Increase ε (capped).
- If $\Delta EMA_{acc} < -\tau_{acc}$: Decrease ε (bounded).
- If $\Delta EMA_{mix} > \tau_{mix}$: Increase λ_{mix}, β_a, β_b (capped).
- If $\Delta EMA_{mix} < -\tau_{mix}$: Decrease λ_{mix}, β_a, β_b (bounded).

This adaptive mechanism allows the regularization strategy to self-tune during training, potentially leading to better performance and reduced sensitivity to initial hyperparameter settings.

3.4 Final Training Objective

The final loss function $\mathscr{L}$ for VectorShake combines the standard cross-entropy loss with the weighted sum of the five latent space regularization losses:

$$\mathscr{L} = \mathscr{L}_{CE} + \lambda_{rec}\mathscr{L}_{rec} + \lambda_{con}\mathscr{L}_{con} + \lambda_{mix}\mathscr{L}_{mix} + \lambda_{cons}\mathscr{L}_{cons} \tag{3.13}$$

where $\lambda_{rec}, \lambda_{con}, \lambda_{cons}$ are hyperparameters balancing the contribution of each regularization term, typically fixed after hyperparameter optimization. Note that λ_{mix} is adjusted dynamically by the DAC. The model parameters θ_e, θ_c and θ_{rec} are updated by minimizing this combined loss $\mathscr{L}$ using an optimizer like AdamW [22].

Chapter 4

Experiments

This section details the experimental setup designed to evaluate the effectiveness and efficiency of VectorShake in few-shot text classification scenarios. We describe the datasets, evaluation metrics, baseline models, and implementation details for VectorShake.

4.1 Datasets and Few-Shot Setup

We evaluate our proposed method and baselines on three diverse English text classification datasets commonly used in NLP research:

- **AG News:** A topic classification dataset consisting of news articles categorized into 4 classes (World, Sports, Business, Sci/Tech). It is widely used for benchmarking text classification models.
- **GoEmotions:** A multi-label emotion classification dataset based on Reddit comments, annotated with 27 emotion categories plus a neutral category. Following common practice for multi-class conversion, we use the primary label provided in the dataset.
- **TREC-6:** A question classification dataset focused on classifying questions into 6 coarse-grained categories (Abbreviation, Entity, Description, Human, Location, Numeric Value). It tests finer-grained semantic understanding.

For each dataset, we simulate few-shot learning scenarios by creating training sets with K examples per class, where K ranges from 1 to 10. This setup is derived from the experimental procedure outlined in our code (‘vectorshaking6.py‘). Specifically, for a given K

and dataset, we randomly sample K training instances for each class using a fixed random seed (42) to ensure reproducibility. The standard test splits provided with each dataset are used for evaluation. The test were conducted on the same seed every time to ensure consistency and reproductibility.

4.2 Evaluation Metrics

We evaluate model performance using standard classification metrics derived from the predictions on the respective test sets:

- **Accuracy:** The primary metric, representing the overall proportion of correctly classified instances.
- **Macro F1-Score:** The unweighted mean of the F1-scores calculated for each class, providing a balanced measure that accounts for potential class imbalance.
- **Macro Precision and Macro Recall:** The unweighted means of precision and recall across all classes, offering further insight into classification performance per class.

In addition to classification performance, we also measure and report the Total Training Time for each method under identical hardware conditions to assess computational efficiency.

4.3 Baselines

We compare VectorShake against two strong baselines:

- **BERT Fine-tune:** A standard BERT model ('bert-base-uncased') fine-tuned directly on the K-shot training data using the standard cross-entropy objective $\mathscr{L}_{CE}$. We use the AdamW optimizer with a learning rate of 2e-5 and weight decay of 0.01, training for 30 epochs, consistent with the baseline training function in our code (‘vectorshake.py‘).
- **SetFit:** A competitive few-shot learning method [2] based on contrastive fine-tuning of Sentence Transformers. We use the implementation provided by the ‘setfit‘ library, employing the ‘sentence-transformers/all-MiniLM-L6-v2‘ model as the

backbone. SetFit is trained on the same K-shot datasets for 30 epochs / 20 iterations following its recommended training procedure.

4.4 VectorShake Implementation Details

Table 4.1: Optimal Hyperparameters found by Optuna for VectorShake on the AG News dataset with $K = 5$ examples.

Hyperparameter	Optimal Value
λ_{mix}	0.6736
λ_{con}	0.0690
λ_{cons}	0.9293
λ_{rec}	0.1933
β_a	0.5202
β_b	3.7447
ε	0.0589
lr	0.0001
weight_decay	0.0034
η_{DAC}	0.4078
τ_{acc}	0.0191
τ_{mix}	0.0009
α_{ema}	0.6209

Our VectorShake implementation, detailed in our code (‘vectorshake.py‘), uses ‘bert-base-uncased‘ as the backbone encoder. The model is trained end-to-end by minimizing the combined loss function (Eq. (3.13)). We use the AdamW optimizer [22]. Key hyperparameters specific to VectorShake, including the initial latent adversarial perturbation magnitude (ε), the mixup Beta distribution parameters (β_a, β_b), the loss weighting coefficients ($\lambda_{rec}, \lambda_{con}, \lambda_{mix}, \lambda_{cons}$), and the DAC parameters ($\eta_{DAC}, \tau_{acc}, \tau_{mix}, \alpha_{ema}$), along with the optimizer's learning rate and weight decay, were tuned using the Optuna framework [23] over a number of trials (e.g., 50 trials as per ‘vectorshaking6.py‘ or 100 trials as mentioned in the original paper text), maximizing accuracy, as implemented in the hyperparameter search functions within our code (‘vectorshake.py‘). The best hyperparameter configuration found during this search was used for the final results reported in this paper. We provide an example of the hyperparameter optimization results on AG News (K=5) in table 4.1.

All experiments were conducted using a batch size of 16, a maximum sequence length of 128 tokens, and trained for 30 epochs with a fixed random seed of 42 for reproducibil-

ity. Experiments were conducted using one NVIDIA RTX 4080 SUPER GPU. Key libraries included PyTorch 2.4.1+cu124, Transformers 4.45.2, Datasets 3.3.2, SetFit (commit 7a25ff9), and Optuna 4.2.1.

Chapter 5

Results

Table 5.1: Accuracy comparison for all datasets and K-shot values. The highest accuracies per column per dataset are bolded.

Model	K=1	K=2	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10
AG News										
BERT Baseline [1]	0.372	0.624	0.665	0.698	0.755	0.784	0.796	0.799	0.794	0.802
SetFit [2]	0.460	0.671	0.690	0.767	0.684	0.776	0.776	0.793	0.757	0.754
VectorShake (ours)	**0.463**	**0.688**	**0.704**	**0.793**	**0.794**	**0.830**	**0.823**	**0.823**	**0.832**	**0.826**
GoEmotions										
BERT Baseline [1]	0.057	0.065	0.084	0.119	0.142	0.171	0.191	0.213	0.221	0.231
SetFit [2]	0.046	0.090	0.114	0.167	0.172	0.211	0.228	0.238	0.255	0.274
VectorShake (ours)	**0.093**	**0.108**	**0.180**	**0.296**	**0.243**	**0.296**	**0.296**	**0.300**	**0.296**	**0.295**
TREC-6										
BERT Baseline [1]	0.300	0.452	0.520	0.572	0.612	0.650	0.674	0.686	0.740	0.740
SetFit [2]	0.406	**0.636**	**0.676**	**0.740**	**0.848**	**0.810**	**0.820**	**0.846**	0.776	**0.812**
VectorShake (ours)	**0.452**	0.560	0.610	0.664	0.744	0.736	0.742	0.746	**0.798**	0.786

This section presents the empirical results comparing VectorShake against the BERT Fine-tune baseline and SetFit on the AG News, GoEmotions, and TREC-6 few-shot text classification tasks for K values ranging from 1 to 10 examples per class. We report Accuracy and Total Training Time in Tables 5.1 and 5.2.

Our results indicate that VectorShake is superior on AG News and GoEmotions compared to both the BERT Baseline and SetFit, while on TREC-6 our results indicate a slightly inferior but still competitive performance, strongly outperforming the baseline.

In terms of training time (excluding evaluations on the full validation set), the BERT Baseline training takes approximately 1% of the SetFit training time, however, its test accuracies consistently fall behind the performance of the two few-shot learning meth-

Table 5.2: Training Time comparison (seconds) for all datasets and K-shot values.

Model	K=1	K=2	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10
AG News										
BERT Baseline [1]	15.5	12.2	12.6	13.2	14.4	14.7	14.7	15.2	16.0	16.3
SetFit [2]	566.7	568.8	577.6	590.3	597.2	606.8	617.5	628.2	646.5	650.8
VectorShake (ours)	12.0	12.5	13.4	13.6	16.4	16.6	17.3	18.4	22.4	21.9
GoEmotions										
BERT Baseline [1]	10.7	14.9	17.6	20.7	25.5	29.4	31.0	35.3	40.8	44.7
SetFit [2]	3631.5	3684.4	3758.7	3798.4	3879.5	3775.9	3934.0	4021.4	4149.6	4300.8
VectorShake (ours)	12.9	24.6	39.7	51.1	72.6	97.3	127.9	156.0	191.5	229.4
TREC-6										
BERT Baseline [1]	2.6	3.0	4.5	4.9	5.1	6.2	7.2	7.9	8.8	8.6
SetFit [2]	356.1	382.0	386.2	447.5	457.7	468.7	482.7	492.1	505.5	523.2
VectorShake (ours)	2.6	3.1	6.5	6.5	7.7	10.8	11.7	12.7	17.4	18.9

ods. Meanwhile, VectorShake achieves its superior performance in just 2.5% of SetFit's training time, deeming it a more time and cost-efficient alternative.

Chapter 6

Ablation Study

To understand the individual contributions of the different components within the VectorShake framework and quantify the impact of the Dynamic Adaptive Curriculum (DAC), we conducted a series of ablation experiments. We systematically disabled key components of VectorShake and evaluated the resulting performance changes.

6.1 Setup

The ablation study was performed on AG News with $K = 5$ and TREC-6 with $K = 5$. We evaluated scenarios with DAC enabled (DAC ON) and disabled (DAC OFF). Starting from the full VectorShake model ("Base"), we created variants by disabling one component at a time: setting $\lambda_{mix} = 0$ (-Mix), $\lambda_{con} = 0$ (-Contrast), $\lambda_{cons} = 0$ (-Consistency), $\lambda_{rec} = 0$ (-Reconstruct), or $\varepsilon = 0$ (-Epsilon). We report Accuracy in Table 6.1.

6.2 Results and Analysis

The ablation study results, presented in Table 6.1, offer several insights into the VectorShake framework.

The impact of individual components varies across datasets and DAC states. For AG News (K=5) with DAC OFF, removing the adversarial perturbation ($\varepsilon = 0$) led to a slight improvement (+0.003), while disabling other components resulted in performance degradation, most notably for reconstruction ($\downarrow$0.031) and consistency ($\downarrow$0.015). With DAC ON for AG News, disabling any single component led to a decrease in accuracy, with reconstruction showing the largest drop ($\downarrow$0.047) followed by consistency ($\downarrow$0.026).

Table 6.1: Ablation study results (Accuracy) on AG News ($K = 5$) and TREC-6 ($K = 5$). Δ indicates change relative to the corresponding Base model.

	AG News ($K = 5$)		TREC-6 ($K = 5$)	
Condition	DAC OFF	DAC ON	DAC OFF	DAC ON
Base (Full Model)	0.783	**0.794**	0.604	**0.744**
-Mix ($\lambda_{mix} = 0$)	0.779 ($\downarrow$0.004)	0.784 ($\downarrow$0.010)	0.714 ($\uparrow$0.110)	0.698 ($\downarrow$0.046)
-Contrast ($\lambda_{con} = 0$)	0.782 ($\downarrow$0.001)	0.790 ($\downarrow$0.004)	0.576 ($\downarrow$**0.028**)	0.584 ($\downarrow$**0.160**)
-Consistency ($\lambda_{cons} = 0$)	0.768 ($\downarrow$0.015)	0.768 ($\downarrow$0.026)	0.710 ($\uparrow$0.106)	0.648 ($\downarrow$0.096)
-Reconstruct ($\lambda_{rec} = 0$)	0.752 ($\downarrow$**0.031**)	0.747 ($\downarrow$**0.047**)	0.688 ($\uparrow$0.084)	0.712 ($\downarrow$0.032)
-Epsilon ($\varepsilon = 0$)	0.786 ($\uparrow$0.003)	0.792 ($\downarrow$0.002)	0.586 ($\downarrow$0.018)	0.584 ($\downarrow$**0.160**)
DAC Impact (ON vs OFF Base)	+0.011		+0.140	

For TREC-6 (K=5) with DAC OFF, the results were markedly different: removing mixup ($\lambda_{mix} = 0$), consistency ($\lambda_{cons} = 0$), or reconstruction ($\lambda_{rec} = 0$) surprisingly led to significant performance improvements (+0.110, +0.106, and +0.084 respectively). Disabling adversarial perturbation or contrastive loss caused a decrease in performance. This suggests that without the DAC's adaptive balancing, certain regularization components can be detrimental on TREC-6 in a low-data (K=5) scenario. However, with DAC ON for TREC-6, removing any component resulted in a performance drop. The largest degradations were observed when removing contrastive loss ($\downarrow$0.160) or adversarial perturbation ($\downarrow$0.160), followed by consistency loss ($\downarrow$0.096). This highlights the DAC's role in effectively orchestrating these regularizers.

The most notable finding is the impact of the Dynamic Adaptive Curriculum (DAC). For AG News (K=5), enabling the DAC resulted in an accuracy improvement of +1.1 percentage points (from 0.783 to 0.794) over the base model with fixed hyperparameters. For TREC-6 (K=5), the DAC provided a more substantial boost of +14.0 percentage points (from 0.604 to 0.744). This underscores the DAC's capability to significantly enhance performance, particularly for challenging datasets like TREC-6 in few-shot settings, by dynamically tuning the regularization strategy. The data also indicates that DAC generally improves or maintains performance even when individual components are ablated, by comparing the 'DAC ON' versus 'DAC OFF' columns for each condition.

In summary, while the individual contributions of latent regularization components are context-dependent and can interact complexly with the dataset and the DAC state, the Dynamic Adaptive Curriculum itself stands out as a consistently beneficial mechanism for improving few-shot text classification performance.

Chapter 7

Discussion and Analysis

Our experiments demonstrate that VectorShake presents a compelling approach for few-shot text classification, considering both effectiveness and efficiency. The results and insights from ablation allow for a deeper analysis.

7.1 Performance Synthesis and Dataset Interactions

VectorShake achieved its strongest results on AG News, consistently outperforming baselines. This suggests the latent regularization combo and DAC suit topic classification well in low data. Performance varied elsewhere. On TREC-6, SetFit achieved higher accuracy, though VectorShake improved significantly over BERT and was much faster. This divergence might stem from task differences: AG News requires broader feature robustness (benefiting VectorShake), while TREC-6 needs finer semantic matching (potentially favoring SetFit's contrastive approach). GoEmotions was challenging; VectorShake showed competitive peaks but more variance. The optimal few-shot approach can be task-dependent; VectorShake excels in robust generalization, while SetFit's strength might be specific semantic matching.

7.2 Insights from Ablation Studies

Ablations provide crucial insights. The most significant is the DAC's consistent, substantial benefit (+5.3 and +8.0 accuracy points vs fixed base), highlighting the value of dynamic regularization tuning in few-shot settings, potentially reducing hyperparameter sensitivity. The latent consistency loss proved critical across settings. Adversarial pertur-

bation and mixup showed context-dependent importance, suggesting DAC might adaptively balance regularizers or that optimal combinations vary by task/data. Reconstruction and contrastive losses contributed positively but more modestly. Overall, while components contribute, the DAC is vital for orchestrating them effectively.

7.3 Efficiency Considerations

VectorShake's efficiency is a key advantage. Training times were significantly lower than SetFit's across all datasets (often >10x faster), while only moderately longer than the BERT baseline. SetFit's sentence-pair generation is costly. VectorShake avoids this by applying latent-space operations during standard mini-batch training, making it more practical for resource-constrained scenarios or rapid experimentation.

7.4 Limitations and Future Work

This study has limitations. Experiments focused on English: multilingual evaluation is needed. The complex interactions and DAC mechanisms warrant deeper investigation beyond Optuna tuning. Comparison to a broader range of recent few-shot methods (e.g., prompt-based) would provide further context.

Future work includes applying VectorShake to different base models and tasks. Combining VectorShake with PEFT methods like LoRA [24] could yield high performance with minimal overhead. Refining the DAC's adaptation rules or exploring theoretical underpinnings of the combined latent regularizations are also promising directions.

Chapter 8

Conclusion

In this paper, we addressed the challenge of achieving both high accuracy and computational efficiency in few-shot text classification. We introduced VectorShake, a novel training framework enhancing BERT fine-tuning by integrating multiple latent-space regularization techniques (reconstruction, adversarial, contrastive, consistency, mixup). Complementing these, VectorShake features a Dynamic Adaptive Curriculum (DAC) that automatically adjusts regularization hyperparameters. Our empirical evaluation on AG News, GoEmotions, and TREC-6 showed VectorShake achieves competitive accuracy, notably on AG News, with significant training time advantages over SetFit. Comprehensive ablation studies confirmed positive component contributions and demonstrated substantial, consistent performance gains from the DAC. The results highlight the potential of combining targeted latent-space regularization with adaptive mechanisms for robust few-shot learning. VectorShake presents a promising and efficient framework for fine-tuning LLMs when labeled data is scarce.

Bibliography

[1] Jacob Devlin et al. “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)*. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[2] Lewis Tunstall et al. “Efficient Few-Shot Learning Without Prompts”. In: *arXiv preprint arXiv:2209.11055* (2022). arXiv: 2209.11055.

[3] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”. In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1126–1135.

[4] Jake Snell, Kevin Swersky, and Richard S. Zemel. “Prototypical Networks for Few-shot Learning”. In: *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. 2017, pp. 4077–4087.

[5] Flood Sung et al. “Learning to Compare: Relation Network for Few-Shot Learning”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2018, pp. 1199–1208. DOI: 10.1109/CVPR.2018.00131.

[6] Tianyi Zhang et al. “Revisiting Few-shot Text Classification: The Role of Data Augmentation”. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2021, pp. 1478–1492. DOI: 10.18653/v1/2021.naacl-main.117.

[7] Timo Schick and Hinrich Schütze. “Exploiting Cloze Questions for Few-Shot Text Classification and Natural Language Inference”. In: *Proceedings of the*

16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume. Association for Computational Linguistics, 2021, pp. 255–269. DOI: 10.18653/v1/2021.eacl-main.20.

[8] Tianyu Gao, Adam Fisch, and Danqi Chen. "Making Pre-trained Language Models Better Few-shot Learners". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, 2021, pp. 3816–3830. DOI: 10.18653/v1/2021.acl-long.295.

[9] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *International Conference on Learning Representations, ICLR 2015*. 2015.

[10] Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. "Adversarial Training Methods for Semi-Supervised Text Classification". In: *International Conference on Learning Representations, ICLR 2017*. 2017.

[11] Chen Zhu et al. "FreeLB: Enhanced Adversarial Training for Natural Language Understanding". In: *International Conference on Learning Representations, ICLR 2020*. 2020.

[12] Haoming Jiang et al. "SMART: Robust and Efficient Fine-tuning for Pre-trained Natural Language Models through Adversarial Training". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020, pp. 6747–6761. DOI: 10.18653/v1/2020.acl-main.597.

[13] Qizhe Xie et al. "Unsupervised Data Augmentation for Consistency Training". In: *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*. 2020, pp. 6256–6268.

[14] Kihyuk Sohn et al. "FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence". In: *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*. 2020, pp. 596–608.

[15] Hongyi Zhang et al. "mixup: Beyond Empirical Risk Minimization". In: *International Conference on Learning Representations, ICLR 2018*. 2018.

[16] Vikas Verma et al. “Manifold Mixup: Better Representations by Interpolating Hidden States”. In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 6438–6447.

[17] Pascal Vincent et al. “Extracting and Composing Robust Features with Denoising Autoencoders”. In: *Proceedings of the 25th International Conference on Machine Learning (ICML ’08)*. ACM, 2008, pp. 1096–1103. DOI: 10.1145/1390156.1390294.

[18] Raia Hadsell, Sumit Chopra, and Yann LeCun. “Dimensionality Reduction by Learning an Invariant Mapping”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Vol. 2. IEEE Computer Society, 2006, pp. 1735–1742. DOI: 10.1109/CVPR.2006.100.

[19] Ting Chen et al. “A simple framework for contrastive learning of visual representations”. In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1597–1607.

[20] Yoshua Bengio et al. “Curriculum learning”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 41–48. DOI: 10.1145/1553374.1553380.

[21] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. arXiv: 1412.6980.

[22] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. 2019. arXiv: 1711.05101.

[23] Takuya Akiba et al. “Optuna: A Next-generation Hyperparameter Optimization Framework”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 2623–2631. DOI: 10.1145/3292500.3330701.

[24] Edward J. Hu et al. “LoRA: Low-Rank Adaptation of Large Language Models”. In: *arXiv preprint arXiv:2106.09685* (2021). arXiv: 2106.09685.