Blending static and spectrally reduced contextual word information improves accuracy and reproducibility in low-resource NLP

Anonymous ACL submission

Abstract

Trade-offs among counteractive objectives, including high accuracy, data scarcity, and insufficient computing resources, are crucial in low-resource Natural Language Processing (NLP). The reproducibility issue of model performance aggravates the challenge further. In this paper, we propose a multifaceted solution that includes: i) a newly designed word embedding, namely the blended embedding, ii) CNN/RNN bi-layer core data-scarcity and low-resource computing, and iii) a CPU-trainable transformer-based NLP pipeline. Precisely, the blended embedding combines spectrally reduced contextual information with a processed, modified static embedding, resulting in improved performance at a reduced embedding dimension. The new embedding also improves reproducibility in downstream NLP tasks. Interrogating the training dynamics, we found that blended embedding minimizes the local structure of the perplexity measure of the trained samples. It also reduces hard-to-learn samples by minimizing the variability of the training dynamics. We also identify a CNN/RNN bi-layer core, optimized against energy cost, achievable accuracy, and CO_2 emissions, which outperforms miniature BERT variants. Further facilitating low-resource computing, the proposed CPU-trainable NLP pipeline leverages the multi-core CPU architecture to efficiently extract contextual information from BERT. Applied to several GLUE benchmark datasets with artificial data scarcity, the proposed NLP pipeline achieves improved accuracy and reproducibility (by 3 to 4x) compared to miniature BERT models, making it pertinent for low-resource NLP applications.

1 Introduction

A trade-off between computational cost and model accuracy is crucial for the design and deployment of NLP solutions in resource-constrained environments (Han et al., 2015; Strubell et al., 2019; Lin et al., 2020). For instance, many NLP tasks suffer from data scarcity, and in other contexts, the model deployment environment lacks memory-intensive, high-performing devices such as GPUs/TPUs, making the trade-off a necessity. Specifically, voice-controlled task completion in healthcare or other applications requires accurate and reproducible model performance, even in the absence of high-performance computing devices. Thus, any NLP solutions for the above applications need multifaceted tuning of models, including memory footprint, CPU-trainability of models, and energy budget (CO_2e footprint) of model design and development. As observed, most State-of-the-Art (SOTA) NLP applications rely on memory-intensive transformer-based implementations in original or miniature forms. The SOTA models range in size from a few hundred megabytes to gigabytes (Devlin et al., 2018; Brown et al., 2020; Strubell et al., 2019) and are suitable for running in environments with limited memory, computational resources, and data scarcity. Additionally, SOTA models rely on large training datasets, which are infeasible for languages with insufficient speakers or limited online resources (Joshi et al., 2020).

Attempts made so far to address the multifaceted challenges of low-resource settings have mostly focused on either addressing data scarcity through data augmentation (Xia et al., 2019; Ragni et al., 2014), reducing the embedding dimension, or creating miniature versions of transformer-based models, such as TinyBERT, DistilBERT, and MobileBERT. Although these approaches stand on their own merits, in a context-dependent manner, a concerted solution addressing accuracy, computational cost, and model deployability remains exclusive and unexplored. Further complicating matters, reproducibility- a common concern in NLP models (Summers and Dinneen, 2021; D'Amour et al., 2022), possesses additional challenges in low-resource settings. In this paper, we address these

problems and provide a solution that comprises

- a newly designed blended embedding equally improves reproducibility for small and large datasets in NLP. It also achieves improved accuracy even at a low embedding dimension, reducing the memory footprint
- Bi-layer CNN/RNN cores and their variants; optimized against CO_2e, energy budget, and accuracy. With the blended embedding, the bi-layer core outperforms miniature BERT variants in low-resource settings
- an optimally tuned CPU-trainable BERT pipeline harnessing the multi-core CPU and multi-threading.

An aspect of low-resource NLP is the limited memory, and the word embedding dimension needed for high accuracy exacerbates this further. For instance, as reported, a 300-dimensional embedding vector, similar to GloVe, consumes approximately 6 GB of memory on a 64-bit system for 2.5 million tokens (Ling et al., 2016a). Therefore, a reduced-size word embedding that maintains the accuracy of models is crucial for addressing resource-constrained NLP. Precisely, a low-dimensional word representation costs less in memory footprint. Many earlier works considered quantization of embedding vectors and other algorithmic adjustments to reduce the memory footprint of NLP models (Lam, 2018) (Ling et al., 2016b; Kim et al., 2020). As a case in point, authors in (Shu and Nakayama, 2017; Kim et al., 2020) minimize the number of parameters needed for word representation, considering words as discrete codes. While these methods addressed how optimally the embedding vectors can be compressed, the dimensionality reduction of the vectors, for instance, as extracted from GloVe (Pennington et al., 2014) and word2vec (Mikolov et al., 2013), has not been addressed. Other studies, such as the Post Processing Algorithm (PPA), reduce the dimensionality of static word embedding (Mu et al., 2017a) by subtracting the mean embedding vector (μ) of the vocabulary and produce isotropic embedding. An immediate recursive extension of the PPA applies another PCA step, followed by an additional PPA layer (Raunak et al., 2019). Implementing a multi-stage PPA process improves classification accuracy and allows further flexibility, but remains restricted to static embedding only (Pennington et al., 2014; Mikolov et al., 2013), leaving out contextual word information as used in the SOTA NLP models (Devlin et al., 2018).

In contrast to earlier works, the blended embedding comprises static and contextual word information. Precisely, it includes the static word embedding resources such as GloVe and word2vec, followed by additional processing through PPA steps (Mu et al., 2017a) and its extensions (Raunak et al., 2019). The contextual word information, extracted from BERT (Devlin et al., 2018), undergoes a spectral filtering through a series of steps involving Discrete Fourier Transform (DFT), Gaussian High Pass (GHP) filtering, and PCA for dimensionality reduction. What mathematical functions optimally merge processed static and filtered contextual information, while remaining exclusive? We conduct an extensive search of candidate functions to select those that perform best across different datasets and models.

Additionally, the blended embedding performs well for both small and large datasets in terms of reproducibility and accuracy (D'Amour et al., 2022; Summers and Dinneen, 2021). To understand how blended embedding improves models' performance, we step into the training dynamics of the models and quantify hard-to-learn, ambiguous, and easy-to-learn data using data cartography (Swayamdipta et al., 2020). As observed for a few GLUE-benchmark tasks considering artificial data scarcity, the blended embedding reduces hard-to-learn samples in the trained dataset and pushes those towards ambiguous regions on a confidence-variability plane. As a result, models' generalization to out-of-distribution data improves better (Swayamdipta et al., 2020). Interestingly, such improvement by the blended embedding remains valid for large datasets as well. While the blended embedding partially addresses memory footprint, reproducibility, substantial environmental implications through CO_2 emission, heat dissipation, and, precisely, the energy budget of NLP are also equally concerning. (Strubell et al., 2019; Cai et al., 2020). Immediate remedies are the miniature BERT variants, but at the cost of deteriorated downstream performance, making them impractical in NLP applications that suffer from scarce data resources and are not of sufficiently small size (in the range of 50 MB). As a solution, we perform a network architecture search and identify a CPU-trainable CNN/RNN fusion core optimized for CO_2e, energy budget, and accuracy. Specifically, along with the blended embedding, the CNN+BiLSTM achieves superior accu-

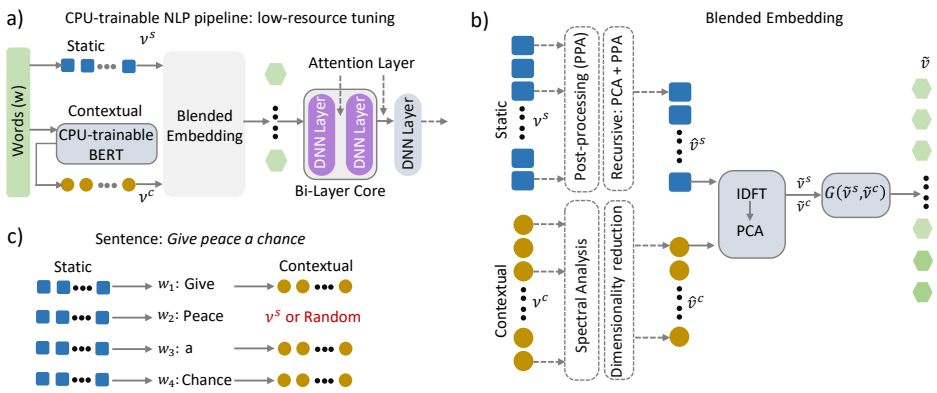


Figure 1: a) Proposed blended embedding model: Both static and contextual embeddings are fused to produce a blended embedding. A bi-layer core with an optional third layer extracts features. b) The blended embedding treats static (through PPA + PCA) and contextual information (through DFT) to produce modified embedding $\widehat{v}^S$ and $\widehat{v}^C$, respectively, and finally a merge function blends both to produce $\tilde{v}$. c) v^s or random values replaces missed $\widehat{v}^C$.

racy and reproducibility compared to the miniature BERT variants. Finally, an efficient CPU-trainable BERT-base implementation harnessing the multi-core structure of CPU provides a fully CPU-trainable NLP pipeline optimized against objectives crucial for resource-constrained NLP applications.

2 Blended Embedding Design

Static and contextual information have their own strengths, and as studied, combining these strengths improves multilingual word representations (Arora et al., 2020; Alghanmi et al., 2020). Interestingly, language structure varies across different levels, ranging from a simple subword level to a document level, with sentential or clause level structure variation in between, imparting a notion of frequency variation, where low frequency corresponds to the document-to-word (or sentence) level, and high frequency corresponds to the word-to-sentence level (Tamkin et al., 2020). A static embedding does not capture words' dependence on their neighboring words (Hofmann et al., 2020). In contrast, contextual embedding captures the context of word meanings within a sentence, making them more informative for various tasks (Devlin et al., 2018; Ethayarajh, 2019). In the blended embedding, we perform a point-wise fusion of the static embedding of a word with its contextual counterpart after both embeddings undergo processing steps designed for dimensionality reduction.

Let us consider that for words ω_i in a sentence vectors $\nu^s(\omega) = \{s_1, s_2, \ldots, s_M\}$ and $\nu^c(w) = \{c_1, c_2, \ldots, c_N\}$ represent the static (S) and contextual (C) embedding, respectively. For a point-wise fusion, both v^s and v^c must be of the same dimension ($M = N$), despite the dissimilar sizes we extract from their sources. So, dimensionality reduction through methods such as PCA or the Autoencoder approach (Hinton and Salakhutdinov, 2006) becomes inevitable. After an initial comparison of alternatives, we chose frequency domain analysis of contextual information over relying solely on PCA or autoencoder-based dimensionality reduction. Spectral analysis (through low-pass, high-pass, and band-pass filtering) of the embedding vector at the neuron level, as seen in earlier studies (Tamkin et al., 2020), separates contextual word information across the sentence, paragraph, and document scales. Earlier works have employed spectral filtering at the neuron level to segregate features hierarchically; here, we apply it to reduce the contextual embedding dimension (Salama et al., 2025b,a), thereby minimizing the memory footprint of word representation. However, we compare alternative approaches, such as the Discrete Fourier Transform (DFT), the Discrete Cosine Transform (DCT), various filter types, and a threshold-based approach targeting the strength of the amplitude spectrum, to determine the optimal method for eliminating weak frequencies. For the processing of static information, we adopted a recursive approach to instill isotropic features in the static embedding (Raunak et al., 2019; Mu et al., 2017b), as outlined in the later sections.

2.1 Spectral processing of contextual embedding

The contextual embedding $\nu^c(w)$ of dimension N is extracted using Global Max Pooling over the

BERT layers. Alternative pooling logic and BERT variants, such as BERT-mini and TinyBERT, have also been explored for extracting contextual information. To spectrally assess $\nu^c(w)$, we compare the Discrete Fourier Transform (DFT) and Discrete Cosine Transform (DCT). The superiority of DFT is evident from the average accuracy over different models and GLUE datasets (Appendix A.12), and we pursue the spectral analysis using DFT as

$$\mathrm{DFT}(V_k) = \sum_{n=0}^{N-1} \nu_n^{(c)} \mathrm{e}^{-\mathrm{j}\,\frac{2\pi}{N}nk} \tag{1}$$

where $V_k \in \{V_0, V_1, \ldots V_{N-1}\}$. For each entry n in ν^c, DFT generates V_k, the Fourier spectrum at frequency element k. The frequency-selective filtering provides a reduced dimension ($\hat{\nu}^c$) (Fig. 1b) based on the magnitude $|V_k|$ of the strength of the sinusoidal component of frequency k. Precisely, we conditioned the magnitude of each spectrum $|V_k|$ to be above a threshold α. The resultant $\widehat{v}^c$ undergoes further dimensionality reduction via PCA or sorting, and assessment reveals PCA outperforms the latter (Appendix A.22). We extract the final contextual information, denoted as $\tilde{\nu}^c$, using the indices obtained from the PCA of $\widehat{v}^c$.

2.2 Post-processing of the static embedding

The static word embedding (ν^s), upon extraction from GloVe (Pennington et al., 2014), undergoes recursive processing for dimensionality reduction. Specifically, we consider the PPA steps (Mu et al., 2017a) and its extension comprises of PCA and an additional PPA step as in (Raunak et al., 2019). A few alternative post-processing methods- i) PPA as in (Mu et al., 2017a), and PCA only, ii) PPA + PCA as in (Raunak et al., 2019), iii) PPA + PCA + PPA + PCA (termed as recursive)- were tried on GLUE-benchmark datasets (QNLI, QQP, SST2, CoLA, SNLI) to identify the optimal post-processing strategy. As seen, the PPA + PCA approach (Raunak et al., 2019) appears superior over the other alternative options, and forms the core post-processing block ν^s stated in Fig. 1b. We also compared different static embedding, where GloVe (Pennington et al., 2014) appears as optimal for computational cost and accuracy (Appendix A.2).

2.3 Static and contextual information fusion

What mathematical process, denoted as $G(\tilde{\nu}^c, \tilde{\nu}^s)$, suffices for an optimal blending of static and contextual information is elusive. We compared linear and non-linear transformations with further variations in the model and datasets, to minimize the context-sensitivity of the potential optimal choice. Specifically, the performance comparison of competing functions includes QQP, QNLI, CoLA, SST2, and SNLI datasets, as well as the three chosen CNN/RNN fusion cores obtained via network architecture search (NAS), which is elaborated on later. The optimal fusion function, $G(\tilde{\nu}^s, \tilde{\nu}^c)$, search is performed against two objectives- i) less power consumption and ii) higher accuracy- to produce a Pareto front (Fig. 3) and select $G(\tilde{\nu}^s, \tilde{\nu}^c)$ for subsequent steps (Appendix A.12, A.14).

3 Low-resource Tuning of NLP Models

3.1 Optimal CNN/RNN stack model search

The miniature BERT-variants (Sanh et al., 2019; Jiao et al., 2019; Sun et al., 2020) are of small size (1.7 to 4.0 MB) but are not deployable on edge devices, and their performance suffers in data scarcity as well. We performed an exhaustive screen of alternative fusion combinations of CNN and RNN modules of length ≤ 3, aiming for performance and energy efficiency. The fusion model stacks the CNN/RNN layers and adds a self-attention layer (Vaswani et al., 2017), generating 105 different models to study. The initial screen utilizes a mean accuracy across QNLI, QQP, and SST2 datasets from the GLUE benchmark (Wang et al., 2018).

3.2 CPU-trainable contextual processing

We employ a multi-core optimization strategy that incorporates data and model-level parallelism, along with measures to minimize bottlenecks in CPU-based BERT training. The process includes data loading parallelism, Intel MKL threading, batch-level parallelism, and memory hierarchy optimization, such as L1/L2 cache hit rates. CPU-trainability analysis uses 1-core, 2-core, 4-core, and 32-core CPU to compare with GPU implementations of BERT variants on SNLI datasets.

4 Experimental Setup and Metrics

4.1 Energy cost and carbon footprint analysis

In default settings, we sampled GPU and CPU power consumption during model training (50 trials for each selection of 500 data points) on a single NVIDIA Tesla T4 and CPU (with a few exceptions, where a NVIDIA Tesla P100 GPU was used in Google Colab). The total power usage in kilowatt-hours (kWh) is formulated as in (Strubell

Model	Blended 75	Context 75	GloVe 75	GloVe 300	Context 300	TinyBERT	DistilBERT	BERT-mini
CNN+LSTM	**62.72**	61.46	62.14	62.51	59.91			
BiLSTM+LSTM	**62.51**	60.56	61.01	62.32	58.33	56.05	61.12	55.64
LSTM+LSTM	61.19	61.24	**61.87**	61.77	59.25			

Table 1: Accuracy comparison between the blended embedding and other approaches. Here, the average of the QQP, QNLI, SST2, CoLA, and SNLI is calculated. The blended embedding performs competitively or outperforms others.

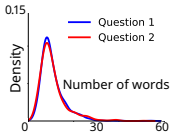
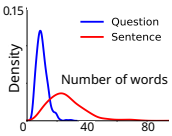
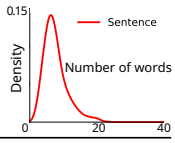
Method	QQP			QNLI			CoLA		
	Acc.	std	Length Distribution	Acc.	std	Length Distribution	Acc	std	Length Distribution
Lowpass	65.09	**0.49**		60.60	1.20		62.20	1.20	
Highpass	64.89	1.02		**61.10**	**0.66**		60.60	**0.58**	
Bandpass	**66.70**	1.21		59.30	**0.75**		**62.89**	1.24	
Threshold	**65.29**	**0.51**		**62.00**	1.48		**62.60**	**0.37**	

Table 2: Showing a comparison among lowpass and highpass (frequencies cutoff at 0.1) and bandpass (a range from 0.05 to 0.3) methods and the length distribution of QQP, QNLI, and CoLA (randomly selected 1000 samples). The threshold refers to a cutoff of 10% of the mean value of the embedding vector, where the blended embedding dimension is n = 75. The mean accuracy (Acc.) and standard deviation (std) were calculated across 5 times.

et al., 2019). Let P_c represent the mean power draw in 50 trials, and the GPU power consumption P_g is accessible from GPU statistics. For a training time t, total power P_t usage is $P_t = 1.58t(P_c + P_g + P_r)/1000$; where P_r is the total power drawn by RAM. The The CO_2 emission calculation follows the formulation $CO_2e = 0.954P_t$ as in Strubell et al. (2019).

4.2 Cartography to explain the role of blended embedding

How blended embedding strengthens models' reproducibility and partly improves their accuracy is studied using data cartography. Precisely, Data cartography assesses training dynamics, employing two metrics — confidence and variability — and produces a data map (Swayamdipta et al., 2020). The data map is used to differentiate data into *easy-to-learn*, *hard-to-learn*, and *ambiguous* based on their projections on the variability-confidence plane. Such region-specific data provides crucial information on the learning and generalization role of data, where ambiguous instances contribute to high performance, easy-to-learn instances aid optimization of model performance, and hard-to-learn instances are often perceived as data errors. The confidence score of a model for an instance m is the mean probability μ_m of the true label (y_i^*) over all training epochs E (Swayamdipta et al., 2020). For detailed formulation, see Appendix A.8.

Algorithm 1 Spectral analysis: word embedding

Require:
Contextual Embedding $\nu^c = \{c_0, c_2, \ldots c_{N-1}\}$
Ensure:
c_i = GlobalMaxPooling($L_i^j, j = 0, 1, \ldots 12$, denotes BERT-layers)
Calculate V_k = DFT(ν^c), μ = mean($|V_k|$)
Identify indices in $|V_k|$, where $|V_k| \geq 0.1\mu, \forall k \in \{1, 2, \ldots N-1\}$
Use these indices to produce a reduced DFT or DCT spectrum, V_k^r
Take n-point PCA (V_k^r, n)
Extract the n-indices of the original DFT spectrum using PCA outcomes
Use these n-indices to retrieve final form of $\tilde{\nu}^c$

5 Results

5.1 Blended embedding improves accuracy and reproducibility

A comparison of newly identified Bi-Layer CNN/RNN cores over multiple datasets shows that blended embedding achieves superior average accuracy as compared to equal or higher dimensions of contextual and static embeddings alone (see Table 1). Such superiority holds against miniature BERT variants by a margin. As in Table 3, in comparison to the miniature BERT-variants for artificial data scarcity, the CNN/RNN core exhibits better accuracy in four (except SNLI) out of the five datasets tried. It is worthwhile noting that blended embed-

Models	QQP	QNLI	SST2	CoLA	SNLI
BERT-base	**71.00**	57.00	81.00	62.00	55.28
DistilBert	64.00	53.60	71.40	58.60	**58.00**
Roberta	54.20	50.40	63.00	63.00	50.20
Bert-Mini	55.00	53.20	69.20	51.40	49.40
mBert	66.40	56.00	67.60	57.40	38.67
TinyBert	59.80	56.40	71.20	50.20	40.67
CL (GloVe)	62.00	64.99	**74.79**	63.00	47.77
BL (GloVe)	**66.50**	63.00	72.19	60.80	49.11
CL (Blend)	65.00	**70.00**	63.19	**69.19**	46.22
BL (Blend)	65.00	68.00	66.59	63.39	**49.55**

Table 3: For SNLI, 1800 samples, and for others, we used 500 randomly selected samples and trained on CNN+LSTM (CL), LSTM+LSTM (LL), and BILSTM+LSTM (BL). Max. accuracy is from 50 trials for the same samples (500), and averaged over 5 different sets. GloVe and blended embedding were 300, 75D

Embeddings with BiLSTM+LSTM Model		Mean Acc.	Standard deviation
SNLI 100%	Glove	72.43	0.60
	Contextual (Frozen)	70.87	0.73
	Blended	72.04	**0.45**
QNLI 100%	Glove	61.45	0.59
	Contextual (Frozen)	62.51	0.96
	Blended	61.69	**0.41**
	BERT-base-uncased	84.90	0.69

Table 4: Reproducibility assessment using Blended embedding for large datasets. The metrics are the average of the 10 and 5 best accuracies on the full SNLI training dataset and the full QNLI dataset, respectively, along with their standard deviations.

		Mean Acc.		std	
Models	Using	500	1000	500	1000
BiLSTM +LSTM	GloVe	59.53	**61.78**	**1.48**	1.32
	MP PCA	60.63	61.63	1.86	1.25
	Blended	**60.82**	**60.83**	**1.27**	**0.70**
CNN +LSTM	GloVe	**62.17**	**62.88**	**0.73**	0.71
	MP PCA	61.57	62.50	2.03	1.02
	Blended	59.89	60.00	1.64	**0.56**
RoBERTa		55.94	78.38	2.91	1.96
BERT-mini		55.72	68.96	3.42	1.78
DistilBERT		57.98	72.21	4.94	2.10
TinyBERT		50.09	61.10	4.47	1.93

Table 5: The models' mean accuracy (over datasets) and reproducibility (standard deviation) were evaluated (from 10 times) with (Blended) and without (Max Pooling PCA (MP PCA)) DFT and compared with GloVe. Embedding dimension used $n = 100$.

ding performs inferiorly for SST2, resulting in a marginal difference in average accuracy reported in 1, mimicking an artificial data scarcity (500 samples). To assess the context-dependency of blended embedding, we analyzed the length distribution of datasets and related it to the role of contextual information inclusion in blended embedding over the static-only word embedding in downstream performance. Considering accuracy and reproducibility as objectives to optimize, alternative filtering through low-, high-, and band-pass reveals context dependency that may relate to the sentential length of the training samples, as shown in Table 2. Instead, we adopt a threshold-based elimination of weaker frequencies that relies solely on the strength of amplitude spectra, thereby eliminating any dependency on parameters such as cut-off frequency or filter choices.

Intriguingly, blended embedding also improves reproducibility, as measured by the standard deviation (std) of N accuracy values of randomly selected train and test samples N times. Averaged over five GLUE datasets, the Bi-Layer core model (BiLSTM+LSTM) with blended embedding achieves a $3-4$ times improvement in reproducibility over alternative BERT variants used in comparison, as shown in Table 5. Such improvement appears to be invariant over sample size (500 and 1000 used) and Bi-Layer core model variation, as has been observed with the CNN + LSTM model (row 2, Table 5). To clarify, the miniature BERT-variants mostly exhibit improved accuracy as the sample size increases to 1000, but despite a reduction in standard deviation, they remain inferior to blended embedding in reproducibility (column 4, Table 5). The improved reproducibility also holds over Glove and BERT-base when the BiLSTM + LSTM model uses blended embedding on 100% of SNLI and QNLI datasets, shown in Table 4.

5.2 Optimal Bi-Layer CNN/RNN core and merging functions

Many earlier works have relied on juxtaposing CNN/RNN models and demonstrated superior performance in various NLP applications (Chiu and Nichols, 2016; Minaee et al., 2021; Maheen et al., 2022; Gupta et al., 2021). Extending beyond, we searched for competing models comprising CNN/RNN layers, considering energy cost and high performance as fusion objectives. Some fusion core models emerge from the search, for instance, BiLSTM + LSTM, LSTM + LSTM, and CNN + LSTM, which outperform BERT-base and its miniature models on several GLUE benchmark datasets, as summarized in Appendix A.11, by mimicking artificial data scarcity. Among the three CNN/RNN cores compared, the CNN + LSTM fu-

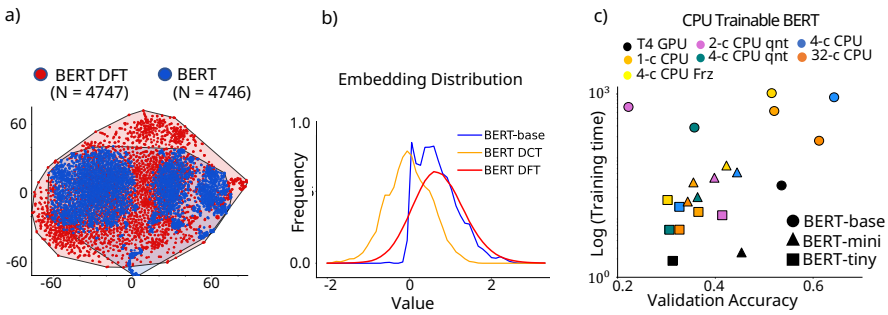


Figure 2: a) Comparison of the extracted BERT-base embedding with/without applying DFT, and others are shown in Appendix A.16. Sample number is N = 4746, 4747 for BERT-base and BERT-base with DFT, respectively. b) DFT is the smoothest distribution among all. c) Trade-off of BERT variants for number of CPU cores and GPU

sion model costs less in power consumption while achieving an accuracy comparable to other fusion models, shown in Fig. 3 (in Appendix 11). These variants of CNN/RNN fusion mostly achieve better accuracy (Table 3) when compared against DistilBert, RoBERTa, mBert, TinyBert, and Bert-mini. As in Table 1, the three CNN/RNN fusion further supports the stated superiority, showing an improvement irrespective of the embedding form. Besides, $G(\tilde{\nu}^c, \tilde{\nu}^s)$ exhibits model dependency. For instance, a constraint of less than 0.02 kWh and accuracy 63 to 68 percent identify $\log(x/y)$ and $x+y$ for the CNN+LSTM model whereas the functions $\log(x/y)$, $(x+y)$, and $\log(x+y)$ appear better-performing blending functions in other models, as in Fig. 3. Also, search for optimal $G(\tilde{\nu}^c, \tilde{\nu}^s)$, appears data sensitive, as summarized in Table 17.

5.3 Blended embedding and CNN/RNN fusion effectively trades off in low-resource

Applications that require edge devices often rely on extensive energy budgeting. The proposed method may be a viable alternative to explore. For instance, a comparison between GloVe and blended embedding, as in Appendix A.3, shows that blended embedding outperforms GloVe accuracy while consuming less energy Additionally, the model options are compared against two performance benchmarks forming objects focused on low-resource NLP in a Pareto-front-like analysis (Fig. 3) to select model options and merger methods for static and contextual information. Such an exploration protocol addresses low-resource NLP, and along with blended embedding, achieves higher accuracy (Table 1, and Appendix A.6), making our work a viable alternative for low-resource NLP.

Among the merging functions and their reproducibility explored in Eq. 2, we employ a simple

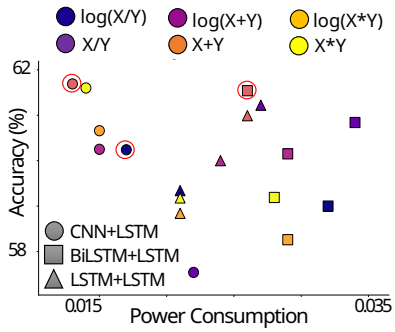


Figure 3: Among the $G(\tilde{\nu}^s, \tilde{\nu}^c)$ alternatives, $x+y$ and $\log(x/y)$ perform better. Accuracy and power consumption are averaged over all datasets considered.

Embedding	Acc (%)	Easy-to-learn (%)	Hard-to-learn (%)	Ambiguous (%)
GloVe 300	**49.18**	49.78	32.00	18.22
BERT DFT 300	48.52	52.56	38.56	8.89
BERT DCT 300	48.96	42.44	52.22	5.33
Blended 75	47.62	**50.11**	**32.22**	**17.67**

Table 6: Comparison with the average percentage of the LL, CL, and BL models' accuracy and data cartography for embeddings using the SNLI dataset.

point-wise addition of the modified static and contextual information. Also, the blended embedding improves performance compared to cases using the static embeddings or miniature variants of BERT, as shown in Table 1 (Appendix A.6). Precisely, the performance of each model improves by a margin of around 1-2% when the blended embedding (size 75) is used instead of the static embedding (size GloVe 300) only. Such a reduction in embedding dimension without compromising accuracy is advantageous from the perspective of memory footprint and energy cost. Compared to miniature BERT models, the fusion models consistently outperform across nearly all instances examined (Table 1).

5.4 Spectral filtering improves generalization in data scarcity

In an investigation focused on how spectral filtering contributes to improved reproducibility, we projected (using t-SNE (Maaten and Hinton, 2008)) the high-dimensional distances among the unique words of the 500 training samples onto a 2D plane. The projection, as shown in Fig. 2.a, demonstrates significant local structures (concentrated blue circles) in the absence of the DFT-based weak frequency elimination. These local structures represent similarity between words and imbue the model with comparatively limited semantic information (Yu et al., 2022; Gong et al., 2019) as opposed to the dispersed version obtained from the DFT-based analysis. In cases of data scarcity, it is crucial to extract more discriminative information from the limited training samples available. The DFT processing provides this by accessing more semantic meaning, as measured through convex hull comparison in Fig. 2a. Additionally, a comparison of the probability density function (PDF) for semantic similarity obtained in alternative approaches reveals a comparatively smoother PDF for DFT-based spectral filtering (see Fig. 2b), which may relate to alleviating overfitting in NLP models.

Using data cartography, we further investigated how blended embedding affects the trained samples and help generalizing more effectively. Data cartography captures training dynamics and quantifies the probability of inferring accurate labels to generate a data map of the training data on a confidence and variability plane, as defined in Eq. 3 and Eq. 4. Among the three subclasses, blended embedding outperforms GloVe and contextual embedding in terms of handling ambiguous data (see Table 6). Ambiguous data helps models treat and learn out-of-distribution data well, enhancing the generalizability of NLP models, as analyzed in (Yang et al., 2023; Swayamdipta et al., 2020). Such performance appears model-independent, as evident from the Bi-Layer CNN/RNN used (Fig. 4) in the data cartography experiment. In model training, easy-to-learn data facilitates tuning and optimization, so its presence among training samples is crucial. In addition to a significant amount of ambiguous data, the blended embedding exhibits comparable easy-to-learn data statistics to those obtained by other approaches. Besides, hard-to-learn data statistics (often treated as label errors in data cartography (Swayamdipta et al., 2020)) shrink when blended embedding encodes words. With its ability to smooth the embedding vectors through DFT-based dimensionality reduction and an increase in ambiguous data, blended embedding contributes to improving generizability and models' reproducibility (Table 4, Table 5).

Task	Setup	Time (s)	GPU/CPU
Training	CPU	171.70	5.4x
	GPU	31.80	
Embedding Extraction (500)	CPU	17.50	5.3x
	GPU	3.30	
Embedding Extraction (1000)	CPU	33.30	4.5x
	GPU	7.40	

Table 7: Performance comparison of training and embedding extraction time using BERT-base-uncased on T4x2 GPU, and our optimized version in a 32-Core CPU

5.5 A CPU-trainable BERT pipeline

To eschew GPU dependency in low-resource environments, we design a CPU-trainable NLP pipeline (Fig. 10) that trains BERT or its variants to extract contextual information necessary for blended embedding. Together with the identified CNN/RNN fusion and blended embedding, the proposed pipeline outperforms (in accuracy, training time, power consumption) the BERT-base model in data scarcity, as shown in Appendix A.7. Precisely, BERT trained using a CPU-trainable pipeline takes only 4 to 5 times longer training time on a 32-core machine (Table 7).

6 Discussion

We combined static and contextual embedding approaches with custom mathematical transformations to form the blended embedding, demonstrating enhanced classification accuracy in our preliminary analysis. Besides, resource-constrained devices often fail to harness SOTA models and may be equipped with stacked CNN/RNN models identified through extensive screening of alternative design choices. Additionally, the models conform to the energy budgeting necessary for resource-constrained devices and consider a low carbon footprint for the underlying computations. Since the spectral analysis only performs selective frequency squeezing to post-process the contextual embedding, we hypothesize that alternative forms of filtering may be worthwhile, considering the frequency dependency of contextual information across various NLP tasks.

Limitations

This study presents a newly devised blended embedding that leverages both static and contextual information to enhance the accuracy of a low-dimensional word representation. The investigation could be extensively applied to other GLUE benchmark datasets and low-end devices for further tuning and adjustments to model size. Moreover, replacing the missing words with static GloVe word vectors gives better performance, as schematically shown in Fig. 1. Identifying an optimal strategy to minimize the occurrence of missing words requires further investigation. In other contexts, such as those involving class imbalance and smaller datasets, their applicability can be thoroughly tested. Additionally, a more consistent sense of uniformity in hardware and computing facilities is necessary during training and running NLP models, which is occasionally compromised due to device switching in the Colab environment.

References

Israa Alghanmi, Luis Espinosa Anke, and Steven Schockaert. 2020. Combining bert with static word embeddings for categorizing social media. In *Proceedings of the sixth workshop on noisy user-generated text (w-nut 2020)*, pages 28–33.

Simran Arora, Avner May, Jian Zhang, and Christopher Ré. 2020. Contextual embeddings: When are they worth it? *arXiv preprint arXiv:2005.09117*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. 2020. Tinytl: Reduce activations, not trainable parameters for efficient on-device learning. *arXiv preprint arXiv:2007.11622*.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the association for computational linguistics*, 4:357–370.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, and 1 others. 2022. Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*, 23(226):1–61.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*.

Zhiqiang Gong, Ping Zhong, and Weidong Hu. 2019. Diversity in machine learning. *Ieee Access*, 7:64323–64350.

Vedika Gupta, Nikita Jain, Shubham Shubham, Agam Madan, Ankit Chaudhary, and Qin Xin. 2021. Toward integrated cnn-based sentiment analysis of tweets for scarce-resource language—hindi. *Transactions on Asian and Low-Resource Language Information Processing*, 20(5):1–23.

Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Valentin Hofmann, Janet B Pierrehumbert, and Hinrich Schütze. 2020. Dynamic contextualized word embeddings. *arXiv preprint arXiv:2010.12684*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the nlp world. *arXiv preprint arXiv:2004.09095*.

Yeachan Kim, Kang-Min Kim, and SangKeun Lee. 2020. Adaptive compression of word embeddings. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 3950–3959.

Maximilian Lam. 2018. Word2bits-quantized word vectors. *arXiv preprint arXiv:1803.05651*.

Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, Song Han, and 1 others. 2020. Mcunet: Tiny deep learning on iot devices. *Advances in Neural Information Processing Systems*, 33:11711–11722.

Shaoshi Ling, Yangqiu Song, and Dan Roth. 2016a. Word embeddings with limited memory. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 387–392.

Shaoshi Ling, Yangqiu Song, and Dan Roth. 2016b. Word embeddings with limited memory. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 387–392.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Syed Mustavi Maheen, Moshiur Rahman Faisal, Md Rafakat Rahman, and Md Shahriar Karim. 2022. Alternative non-bert model choices for the textual classification in low-resource languages and environments. In *Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing*, pages 192–202.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learning–based text classification: a comprehensive review. *ACM computing surveys (CSUR)*, 54(3):1–40.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017a. All-but-the-top: Simple and effective postprocessing for word representations. *arXiv preprint arXiv:1702.01417*.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017b. All-but-the-top: Simple and effective postprocessing for word representations. *arXiv preprint arXiv:1702.01417*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Anton Ragni, Kate M Knill, Shakti P Rath, and Mark JF Gales. 2014. Data augmentation for low resource languages. In *INTERSPEECH 2014: 15th annual conference of the international speech communication association*, pages 810–814. International Speech Communication Association (ISCA).

Vikas Raunak, Vivek Gupta, and Florian Metze. 2019. Effective dimensionality reduction for word embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 235–243.

Rana Salama, Abdou Youssef, and Mona Diab. 2025a. Combining discrete wavelet and cosine transforms for efficient sentence embedding. *arXiv preprint arXiv:2508.00420*.

Rana Aref Salama, Abdou Youssef, and Mona Diab. 2025b. Semantic compression for word and sentence embeddings using discrete wavelet transform. *arXiv preprint arXiv:2508.00220*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Raphael Shu and Hideki Nakayama. 2017. Compressing word embeddings via deep compositional code learning. *arXiv preprint arXiv:1711.01068*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

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

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. *arXiv preprint arXiv:2009.10795*.

Alex Tamkin, Dan Jurafsky, and Noah Goodman. 2020. Language through a prism: A spectral approach for multiscale language representations. *Advances in*

Neural Information Processing Systems, 33:5492–5504.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Mengzhou Xia, Xiang Kong, Antonios Anastasopoulos, and Graham Neubig. 2019. Generalized data augmentation for low-resource translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5786–5796, Florence, Italy. Association for Computational Linguistics.

Linyi Yang, Yaoxian Song, Xuan Ren, Chenyang Lyu, Yidong Wang, Jingming Zhuo, Lingqiao Liu, Jindong Wang, Jennifer Foster, and Yue Zhang. 2023. Out-of-distribution generalization in natural language processing: Past, present, and future. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4533–4559.

Yu Yu, Shahram Khadivi, and Jia Xu. 2022. Can data diversity enhance learning generalization? In *Proceedings of the 29th international conference on computational linguistics*, pages 4933–4945.

A Appendix

A.1 Initial model selection process

The 105 models are classified into three combinations:

- **CNN + BiLSTM + LSTM (39 models):** 27 (3-layer), 9 (2-layer), 3 (1-layer)
- **CNN + Attention + BiLSTM + LSTM (39 models):** Same layer distribution as above, with an attention layer added in the second position
- **CNN + Attention + BiLSTM + Attention + LSTM (27 models):** Adds a second attention layer to the previous combination of a three-layer model

A.2 Recursive post processing with GloVe

	100 D			300 D
Data	$Case1$	$Case2$	$Case3$	$GloVe$
SNLI	48.66	49.03	47.25	**49.18**
CoLA	**65.00**	62.00	63.66	61.00
QQP	60.67	60.00	**63.33**	62.83
SST2	70.33	69.00	71.67	**73.66**
QNLI	64.99	65.33	**65.99**	64.33

Table 8: Comparison of the GloVe reduced 100 dimensions using PCA, Post Processing, and the Recursive Post Processing Algorithm for SNLI, CoLA, QQP, SST2, and QNLI. The average is calculated based on three CNN/RNN models. This experiment implies that following the PCA, the recursive post-processing Algorithm can improve the accuracy with the reduced 100 dimensions GloVe compared to GloVe with 300 D. $Case1 : PPA, \quad Case2 : PPA - PCA - PPA$ $Case3 : PPA - PCA - PPA - PCA - PPA.$

A.3 Power consumption vs Accuracy for GloVe, contextual, and blended embedding

Data	Accuracy			Power Consumption (kWh)		
	Blend 75	GloVe 300	Context 300	Blend 75	GloVe 300	Context 300
QQP	**64.33**	62.33	64.06	**0.0237**	0.0453	0.0430
QNLI	**67.66**	64.33	57.99	0.0407	0.0430	**0.0356**
SST2	66.06	**73.66**	67.19	0.0367	0.0370	**0.0362**
CoLA	**64.93**	61.00	62.00	0.0152	**0.0099**	0.0114
SNLI	47.70	**49.18**	44.59	**0.0202**	0.0244	0.0215

Table 9: Power consumption and accuracy comparison between the blended embedding and GloVe for QQP, QNLI, SST2, CoLA, and SNLI, where blended 75 D embedding outperforms GloVe 300 D in accuracy and power consumption.

A.4 Merging functions for the Blended embeddings

$$\tilde{\nu} = \log(\tilde{\nu}^s + \tilde{\nu}^c),\ \tilde{\nu} = \tilde{\nu}^s + \tilde{\nu}^c,$$
$$\tilde{\nu} = \log(\tilde{\nu}^s \odot \tilde{\nu}^c),\ \tilde{\nu} = \tilde{\nu}^s \odot \tilde{\nu}^c,$$
$$\tilde{\nu} = \log(\tilde{\nu}^s / \tilde{\nu}^c),\ \tilde{\nu} = \tilde{\nu}^s / \tilde{\nu}^c,$$
$$\tilde{\nu} = (\alpha \cdot \tilde{\nu}^s) + ((1-\alpha) \cdot \tilde{\nu}^c),$$
$$\tilde{\nu} = \sqrt{(\tilde{\nu}^{c^2} + \tilde{\nu}^{s^2} + 2 \cdot \tilde{\nu}^c \cdot \tilde{\nu}^s \cdot \cos(\theta))},$$
$$\tilde{\nu} = \text{Concat}(\tilde{\nu}^s, \tilde{\nu}^c),\ \tilde{\nu} = \text{Concat}(\tilde{\nu}^c, \tilde{\nu}^s)$$
$$\text{Here, } \alpha = 0.3, 0.5, 0.7; \theta = 0, 30, 45, 60, 90 \quad (2)$$

A.5 Comparison of merging methods

Methods	CNN+LSTM	BiLSTM+ LSTM	LSTM+ LSTM
$\alpha = 0.3$	47.55	47.99	44.44
$\alpha = 0.5$	**50.22**	49.33	**47.99**
$\alpha = 0.7$	48.44	**50.00**	44.88
$\cos(0)$	47.99	47.77	41.99
$\cos(30)$	46.88	45.55	42.88
$\cos(45)$	46.88	45.33	39.33
$\cos(60)$	46.44	43.77	39.55
$\cos(90)$	46.44	44.22	36.88
DFT	46.22	49.55	47.33
DCT	44.44	49.11	47.33
Concat(^{s}, $\tilde{\nu}^c$)	44.22	47.11	48.66
Concat(^{c}, $\tilde{\nu}^s$)	44.66	47.33	47.33

Table 10: This comparison shows the performance across different blending methods with 75 D over the SNLI dataset with 1800 training samples. The overall performance of the alpha blend for 0.5 yields better accuracy, as indicated by the formulation in Eq. 2.

A.6 Accuracy comparison among the ($\log(x+y)$, $\log(x/y)$) vs. contextual vs. static embedding

Datasets	log(X/Y)	log(X+Y)	Contextual	GloVe
SST2	65.77	**66.26**	67.17	71.51
QNLI	62.17	**64.49**	61.94	64.08
QQP	**62.54**	62.46	64.21	62.54
CoLA	**63.86**	63.45	62.00	61.43

Table 11: Comparison of accuracy between the Blended embedding (log(x+y), log(x/y)) (average of dimension n = 75, 100, 150, and 300) vs. Contextual vs. Static embedding (GloVe-300) and average of the three CNN/RNN models. The blended embedding can achieve a competitive classification score. Here, the SST2, QNLI, QQP, and CoLA datasets are analyzed to get a better blended function.

A.7 Multi-core CPU Threading VS GPU

Approach		Training Time (s)	Validation Accuracy	Trainable Parameters
T4 GPU	BERT-tiny	1.855838	31.25%	4386307
	BERT-mini	2.548604	45.36%	11171331
	BERT-base	31.803269	53.54%	109484547
1-core CPU	BERT-tiny	18.398381	30.21%	4386307
	BERT-mini	67.493614	42.27%	11171331
	BERT-base	1033.272339	51.52%	109484547
4-core CPU	BERT-tiny	14.214952	32.65%	4386307
	BERT-mini	52.114294	44.44%	11171331
	BERT-base	884.328443	64.29%	109484547
4-core CPU Frozen Layers	BERT-tiny	11.747588	36.53%	3989763
	BERT-mini	35.708459	35.51%	8012291
	BERT-base	526.761116	52.04%	45693699
2-core CPU Quantized	BERT-tiny	10.361756	41.41%	4386307
	BERT-mini	42.579481	39.80%	11171331
	BERT-base	613.764228	22.22%	109484547
4-core CPU Quantized	BERT-tiny	5.991767	30.61%	4386307
	BERT-mini	20.505313	36.36%	11171331
	BERT-base	282.028554	35.71%	109484547
32-core CPU	BERT-tiny	6.015957	32.65%	4386307
	BERT-mini	17.371452	34.34%	11171331
	BERT-base	171.744775	61.22%	109484547

Table 12: Comparison of CPU-based BERT model training performance across hardware configurations and optimization techniques. Results demonstrate our proof of concept that multi-core CPU architectures can effectively train transformer models without the need for GPU acceleration. Optimizations, including parallelization, layer freezing, and quantization, significantly reduce training time while maintaining competitive validation accuracy, establishing viable training pathways for resource-constrained GPU-free environments. *Batch Size: 16, In Frozen Bert approach: 9 layers are frozen, In Quantization Approach: FP32 $\rightarrow$ INT8.*

A.8 Dataset Cartography Formulation

The confidence score of a model for an instance m is the mean probability μ_m of the true label (y_i^*) over all training epochs E (Swayamdipta et al., 2020).

$$\hat{\mu}_m = \frac{1}{E} \sum_{n=1}^{E} p_{\theta^{(n)}}(y_m^* \mid \mathbf{x}_m) \tag{3}$$

where $p_{\theta^{(n)}}$ is model's probability with parameters $\theta^{(n)}$ after n epochs. The spread of $p_{\theta^{(n)}}(y_n^* \mid \mathbf{x}_n)$ quantifies the variability, defined as

$$\hat{\sigma}_m = \sqrt{\sum_{n=1}^{E} \left(p_{\theta^{(n)}}(y_m^* \mid \mathbf{x}_m) - \hat{\mu}_m\right)^2 / E} \tag{4}$$

A.9 Dimension reduction approaches

Over the models and datasets Average Acc.						
Dimensions	DFT_PCA	DFT_SORT	DCT_PCA	GHP_PCA	Sent. DFT	BERT-mini DFT
50 D	**60.65**	58.60	58.51	58.31	58.60	58.89
75 D	**61.21**	58.11	58.93	59.29	60.29	58.47
100 D	**60.34**	59.50	58.85	57.92	59.12	57.68
150 D	**61.31**	59.52	59.12	57.16	58.68	58.77
Over the models and dimensions Average Acc.						
Models	DFT_PCA	DFT_SORT	DCT_PCA	GHP_PCA	Sent. DFT	BERT-mini DFT
CNN+LSTM	60.33	58.87	**60.42**	59.13	59.53	60.18
BiLSTM+LSTM	**60.91**	59.38	58.91	57.94	59.75	58.53
LSTM+LSTM	**61.39**	58.55	57.22	57.44	58.23	56.65
std	**0.78**	0.84	1.48	1.26	1.00	1.72

Table 13: Comparing the reduction methods used in the spectral analysis of contextual embeddings from BERT. PCA represents the selection of the desired dimension (n), as in Algorithm 1, done by taking the principal component from the DFT, and the SORT stands for the same, which has been done by sorting the top-n spectrum from V_k^r. Additionally, spectral analysis of contextual embeddings using the DCT, GHP, which allows only the high-frequency spectrum, one-hot sentence encoding with DFT (Sent. DFT), and the embedding from BERT-mini, following the DFT algorithm, is shown. The DFT with the PCA method outperforms all the other approaches.

A.10 Missing Words alternatives

		Contextual 300				
Data	Models	zero init.	random init	with GloVe	with sent2vec	random init with limits
QQP	CNN+LSTM	65.19	**69.99**	70.99	**73.00**	**68.00**
	BiLSTM+LSTM	62.39	**69.99**	**71.00**	**67.00**	70.99
	LSTM+LSTM	64.59	**68.99**	**70.99**	68.99	**68.00**
SNLI	CNN+LSTM	49.11	**49.55**	49.11	**47.99**	45.55
	BiLSTM+LSTM	**47.33**	**48.22**	**48.44**	**48.44**	**48.44**
	LSTM+LSTM	**47.55**	**45.33**	**49.33**	**49.33**	**46.66**
SNLI	CNN+LSTM	48.88	52.44	**54.44**	**0.0254**	0.0307
	BiLSTM+LSTM	50.22	50.88	**54.00**	**0.0398**	0.0451
	LSTM+LSTM	48.22	51.99	**55.77**	**0.0452**	0.0521

Table 14: The report compares the outcomes if the missing words from contextual embedding are replaced with the GloVe, sent2vec static embedding, or random initialization with and without max and min elements of the vector space. The final observation is that missing words replaced by the GloVe generate better outcomes than others, even though it keeps some missing words as zero-initialized.

A.11 Model performance experiment over three datasets

Serial	Models	Mean Accuracy	CO_2 $kgCO_2e$	Energy (kWh)	Inference Time
1	CNN+CNN+CNN	63.33	0.043	0.045	0.45
2	CNN+CNN+BiLSTM	63.99	0.042	0.0438	1.04
3	CNN+BiLSTM+LSTM	66.99	0.046	0.048	2.92
4	BiLSTM+LSTM+CNN	66.99	0.038	0.04	1.34
5	BiLSTM+BiLSTM+BiLSTM	64.99	0.064	0.067	2.15
6	**BiLSTM+LSTM+LSTM**	69.99	0.06	0.07	3.01
7	**LSTM+LSTM**	64.33	0.039	0.04	1.84
8	**CNN+LSTM**	66.99	0.028	0.03	0.495
9	LSTM+CNN	64.99	0.048	0.051	1.09
10	**BiLSTM+LSTM**	66.33	0.065	0.068	1.27
11	CNN+A+BiLSTM+A+CNN	62.33	0.045	0.048	1.66
12	CNN+A+BiLSTM	64.32	0.058	0.06	1.07
13	BiLSTM+A	62.66	0.054	0.0562	1.61
14	CNN+A+LSTM+BiLSTM	66.33	0.069	0.072	1.71
15	BiLSTM+A+CNN+BiLSTM	54.66	0.086	0.09	3.59
16	LSTM+A+LSTM+LSTM	65.33	0.07	0.074	2.31
17	LSTM+A+CNN+CNN	66.67	0.04	0.04	1.08
18	LSTM+A+BiLSTM+BiLSTM	65	0.11	0.10	4.14
19	LSTM+A+BiLSTM+CNN	64.66	0.08	0.08	2.56

Table 15: Comparison between selected models formed by stacking CNN/RNN layers. Mean accuracy, CO_2 emission, energy requirement, and inference time (for 20 samples) are considered for Comparison. These are the average values of QNLI, QQP, and SST2 datasets. Since different GPUs have been used, training times for the models may have changed partially, but the trial counts for the training models have remained consistent. The CoLA dataset was thoroughly evaluated using only a T4 GPU to address potential concerns regarding reproducibility in Table 5. Following previous studies that identified a fusion length of three as optimal for a DNN stacked structure, we expand the search pool by allowing each layer to host a CNN, LSTM, or BiLSTM. Including the self-attention layer (A) between two successive CNN/RNN layers also forms additional model counts, totaling 105 alternative combinations for the initial phase of the model search process (Shown in Appendix A.1). Each model was trained on the QNLI, SST2, and QQP datasets (Wang et al., 2018) and ranked based on the mean accuracy across these three datasets. After the initial screening, a pool of 19 models was studied for subsequent analysis to assess the relationship between accuracy, training time, FLOPS, energy consumption, and carbon footprint (Joshi et al., 2020; Strubell et al., 2019).

A.12 Static and Contextual embedding merging through DFT, DCT performance evaluation

Datasets	Models	log(X/Y)	X/Y	log(X+Y)	X+Y	log(X*Y)	X*Y
				With DFT Blend			
QNLI	CNN+LSTM	61.19	61.79	**64.39**	**64.39**	61.19	62.40
	BiLSTM+LSTM	59.79	63.99	64.99	**66.79**	60.39	61.19
	LSTM+LSTM	62.59	**66.39**	62.39	65.19	63.99	63.99
QQP	CNN+LSTM	62.99	63.59	61.00	62.80	63.20	**65.59**
	BiLSTM+LSTM	63.19	**65.19**	61.80	62.20	61.99	62.60
	LSTM+LSTM	61.80	**64.39**	62.59	63.60	59.40	59.19
SST2	CNN+LSTM	65.39	61.19	64.59	**66.39**	**66.39**	62.99
	BiLSTM+LSTM	**65.39**	65.19	**65.39**	65.19	64.39	63.19
	LSTM+LSTM	65.99	65.99	**66.99**	64.19	**66.99**	65.69
CoLA	CNN+LSTM	**68.99**	60.00	66.40	63.99	**68.99**	68.79
	BiLSTM+LSTM	63.99	63.39	62.79	**66.40**	61.20	63.19
	LSTM+LSTM	63.00	62.00	63.39	**64.19**	59.39	60.80
SNLI	CNN+LSTM	42.66	41.11	44.88	**50.89**	43.55	48.22
	BiLSTM+LSTM	42.66	46.44	45.77	**47.11**	43.33	45.77
	LSTM+LSTM	43.33	47.33	44.66	**47.77**	44.44	46.22
				With DCT Blend			
QNLI	CNN+LSTM	62.99	**64.99**	62.00	62.99	62.99	56.99
	BiLSTM+LSTM	**67.00**	62.99	64.99	62.99	66.00	56.00
	LSTM+LSTM	63.99	62.99	62.99	60.00	**67.00**	60.00
SST2	CNN+LSTM	66.00	58.99	63.99	63.99	63.99	**67.00**
	BiLSTM+LSTM	**67.00**	63.99	66.00	63.99	62.99	62.00
	LSTM+LSTM	**66.00**	60.00	60.00	62.00	64.99	62.00
QQP	CNN+LSTM	**67.00**	62.00	61.00	62.99	63.99	56.99
	BiLSTM+LSTM	**66.00**	64.99	56.99	60.00	64.99	60.00
	LSTM+LSTM	64.99	62.99	60.00	60.00	**66.00**	56.99
CoLA	CNN+LSTM	62.99	**66.00**	**66.00**	64.99	61.00	63.99
	BiLSTM+LSTM	61.00	**64.99**	62.00	62.00	62.00	58.99
	LSTM+LSTM	**64.99**	62.99	62.99	57.99	62.99	58.99
SNLI	CNN+LSTM	44.22	40.00	44.66	**48.44**	41.11	46.00
	BiLSTM+LSTM	48.22	45.33	47.55	**49.11**	44.88	46.66
	LSTM+LSTM	45.55	41.77	**47.33**	46.44	44.44	44.22

Table 16: Comparison of the alternative merging approaches of a word's static and contextual (with DFT and DCT processing) information. Here, x, y denotes $\tilde{\nu}^s$ (static) and $\tilde{\nu}^c$ (contextual) embedding vectors, respectively. Bold fonts are the best-performing mergers.

A.13 Alternative merging approaches of static and contextual information

Contextual Process	Models	log(X/Y)	X/Y	log(X+Y)	X+Y	log(X*Y)	X*Y
DFT	CNN+LSTM	60.24	57.54	60.25	**61.69**	60.66	61.60
	BiLSTM+LSTM	59.00	60.84	60.15	**61.54**	58.26	59.19
	LSTM+LSTM	59.34	**61.22**	60.00	60.99	58.84	59.18
DCT	CNN+LSTM	60.64	58.40	59.53	**60.68**	58.62	58.19
	BiLSTM+LSTM	**61.84**	60.46	59.51	59.62	60.17	56.73
	LSTM+LSTM	**61.10**	58.15	58.66	57.29	61.08	56.44

Table 17: Comparison of the alternative merging approaches of a word's static and contextual information. Here, x, y denotes $\tilde{\nu}^s$ (static) and $\tilde{\nu}^c$ (contextual) embedding vectors, respectively. Bold fonts are the best-performing models, with an average performance across the SNLI, QNLI, QQP, SST2, and CoLA datasets. Two processes (DFT and DCT for contextual embedding processing) are selected after observing the models' performance in all the datasets.

A.14 Blending function analysis over dimensions

Alpha Blend		$\alpha = 0.3$		$\alpha = 0.5$		$\alpha = 0.7$	
Dataset	Models	75	300	75	300	75	300
SNLI	CNN+LSTM	47.55	49.55	**50.22**	49.11	48.44	49.11
	BiLSTM+LSTM	47.99	48.44	**49.33**	47.33	50.00	48.44
	LSTM+LSTM	44.44	46.88	**47.99**	46.88	44.88	44.88

Table 18: Comparison between dimensions n = 75 and n = 300 with three different α values for the SNLI dataset. When $\alpha = 0.5$ for 75D, this blending function performs better.

Concatenate Blend		SC		CS	
Dataset	Models	75	300	75	300
SNLI	CNN+LSTM	44.22	46.88	44.66	**47.55**
	BiLSTM+LSTM	47.11	48.22	47.33	**49.55**
	LSTM+LSTM	48.66	**49.55**	47.33	**49.55**

Table 19: This analysis evaluates two-dimensionalities, n = 75 and n = 300, using two concatenation approaches on the SNLI dataset. The methods combine static embedding vectors (S) with contextual embedding vectors (C) in two configurations: static followed by contextual and reverse order. The configuration results in contextual embeddings (denoted as CS) achieving superior performance with 300-dimensional embeddings.

Cosine Blend		$\theta = 0$		$\theta = 30$		$\theta = 45$		$\theta = 60$		$\theta = 90$	
Dataset	Models	75	300	75	300	75	300	75	300	75	300
SNLI	CNN + LSTM	47.99	46.66	46.88	47.33	46.88	47.11	46.44	48.88	46.44	**49.55**
	BiLSTM + LSTM	**47.77**	44.66	45.55	44.88	45.33	44.44	43.77	44.22	44.22	43.55
	LSTM + LSTM	41.99	**43.33**	42.88	39.33	39.33	42.22	39.55	41.99	36.88	40.66

Table 20: This analysis evaluates two dimensionalities, n = 75 and n = 300, using a cosine formula with different values for θ on the SNLI dataset.

A.15 SNLI Data Cartography, 1800 Samples

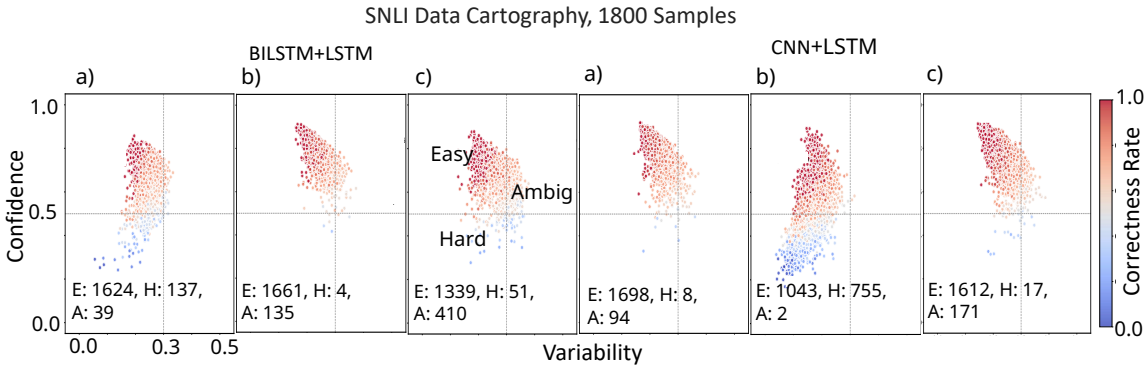


Figure 4: Mapping of trained data as *easy (E)*, *hard (H)* and *ambiguous*: a) GloVe, b) Contextual with DFT, and c) Blended embedding with n = 75 D. Detailed analyses of data mapping are shown in Fig. 9.

A.16 Spectral Distance Analysis of Embeddings

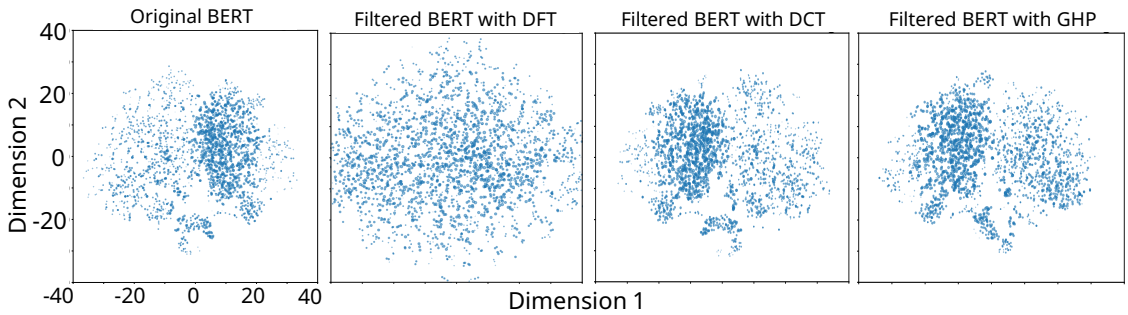


Figure 5: This visual representation shows a comparison among the original BERT-base extracted embedding and the differences after applying the Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), and Gaussian High-pass filter (GHP). The DCT and GHP filters create a sparse, clustered embedding. However, the DFT processing reduces most of the weak frequencies, resulting in a tightly clustered embedding, which indicates a significant reduction in noise/outliers.

A.17 Embedding VS RoBERTa Performance

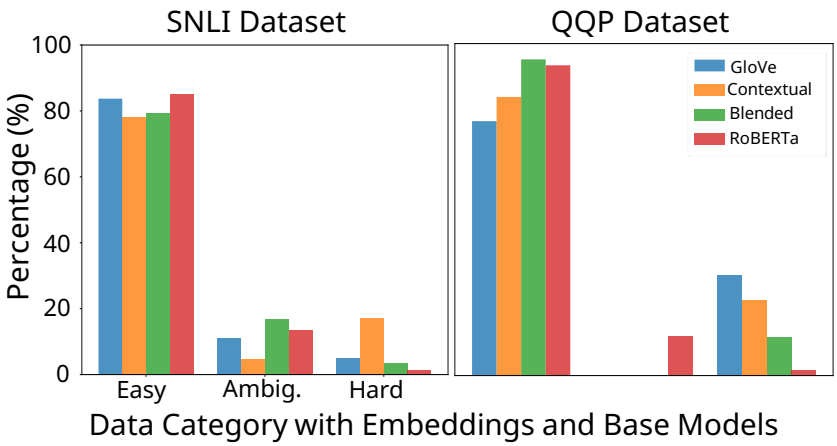


Figure 6: Performance comparison among static, contextual, and blended embeddings is presented for the SNLI and QQP datasets. The average of three CNN/RNN models is taken, revealing that the blended embedding yields a decrease in the hard-to-learn rate, while the easy-to-learn and ambiguous rates increase, utilizing minimum computational resources and approaching the levels of RoBERTa and DistilBERT base models.

A.18 Data Mapping for Base models

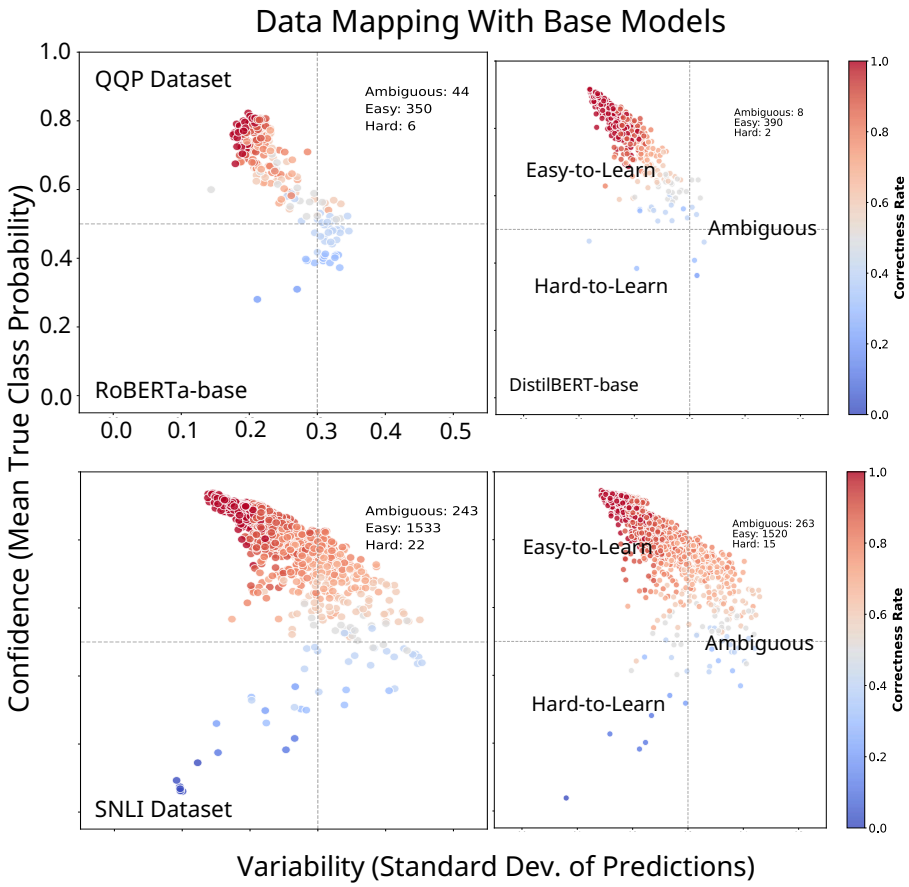


Figure 7: A data mapping of RoBERTa-base and DistilBERT-base models for the QQP and SNLI datasets. Compared to it, QQP and SNLI data mapping, respectively, Fig. 8 and Fig. 9 with the $\alpha = 0.5$ Blended embedding, are performing better in low-resource environments with CNN/RNN models.

A.19 QQP, SNLI Data Mapping with Embeddings

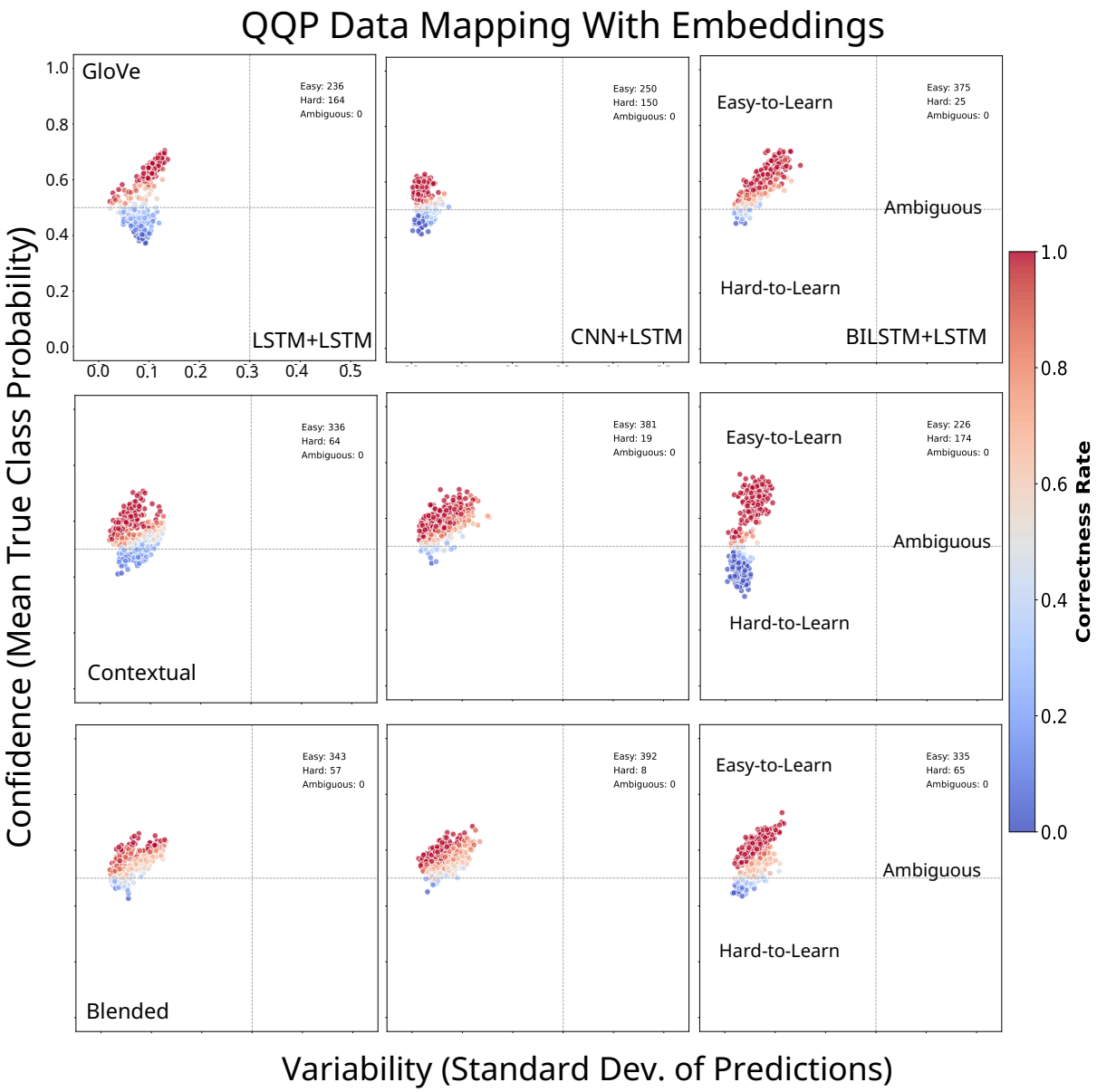


Figure 8: A data mapping comparison among static, contextual, and blended embeddings with three CNN/RNN models for the QQP dataset. In the Blended alpha embedding, the Hard-to-Learn is reduced, and the Easy-to-Learn and Ambiguous are increased. Moreover, the correctness rate of samples is also increased.

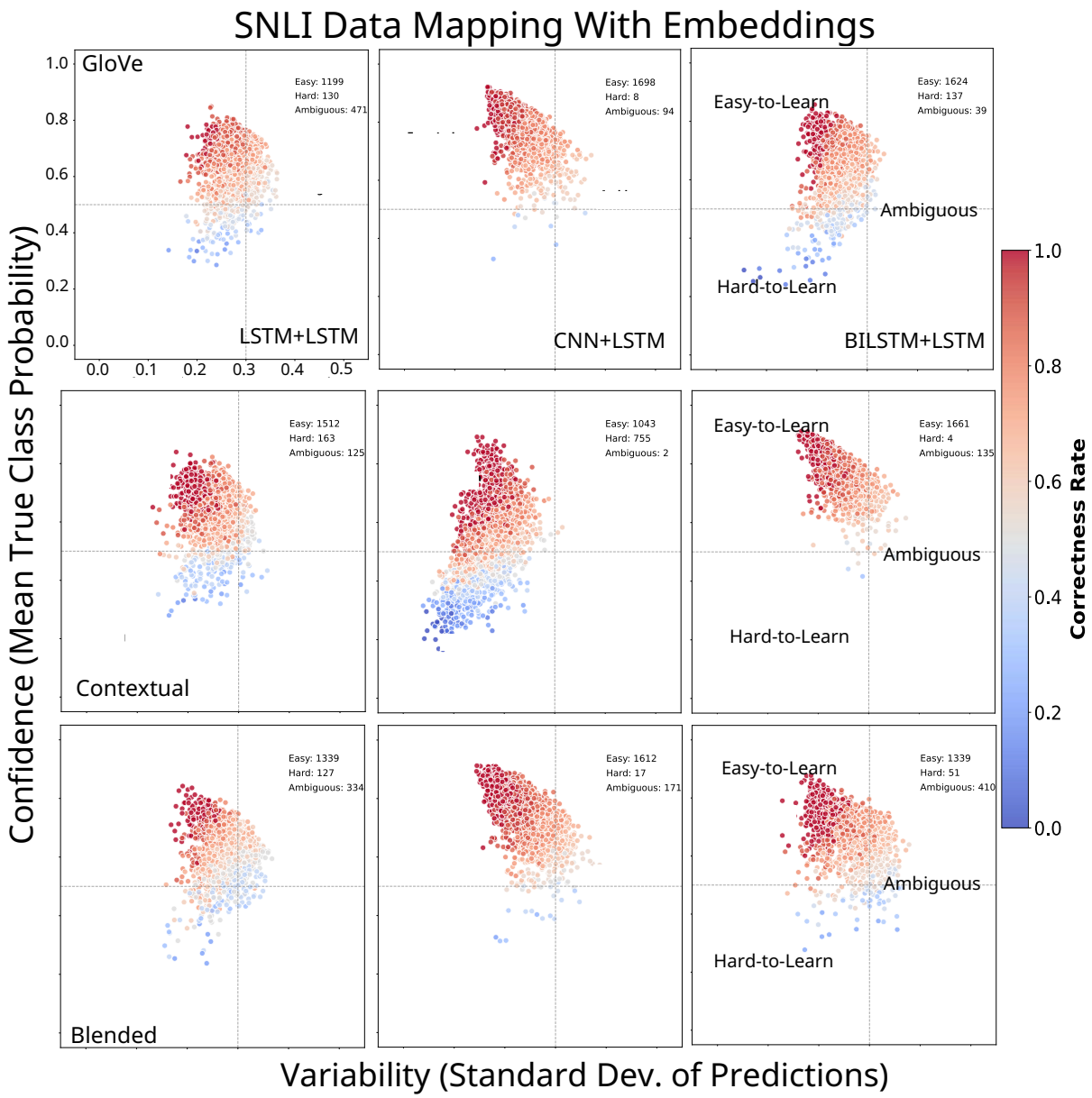


Figure 9: A data mapping comparison among static, contextual, and blended embeddings with three CNN/RNN models for the SNLI dataset. In the Blended alpha embedding, the Hard-to-Learn is reduced, and the Easy-to-Learn and Ambiguous are increased. Moreover, the correctness rate of samples is also increased.

A.20 Multi-Core Threading Pipeline

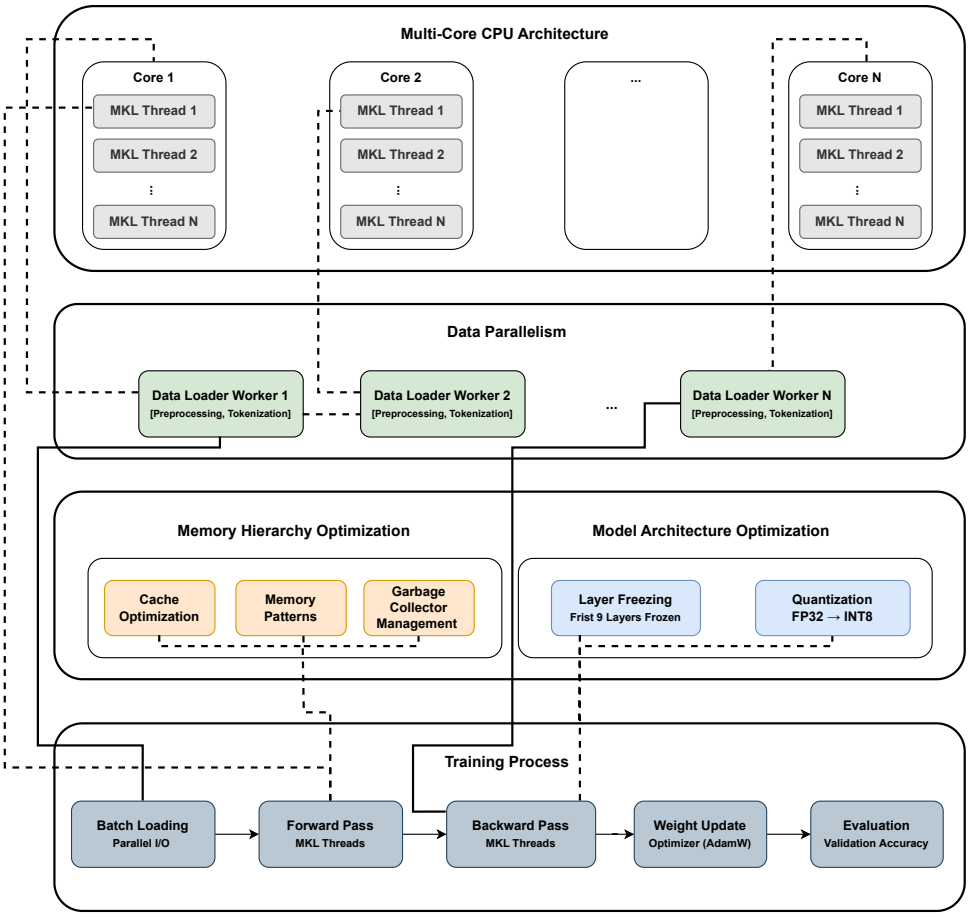


Figure 10: Multi-Core CPU training pipeline utilized parallelized MKL threads per core, concurrent data loading workers for preprocessing, and memory-hierarchy optimizations. The architecture incorporates layer freezing and INT8 quantization techniques to reduce computational requirements while preserving model performance.

A.21 Power and Accuracy comparison among the Blended embeddings with their model sizes

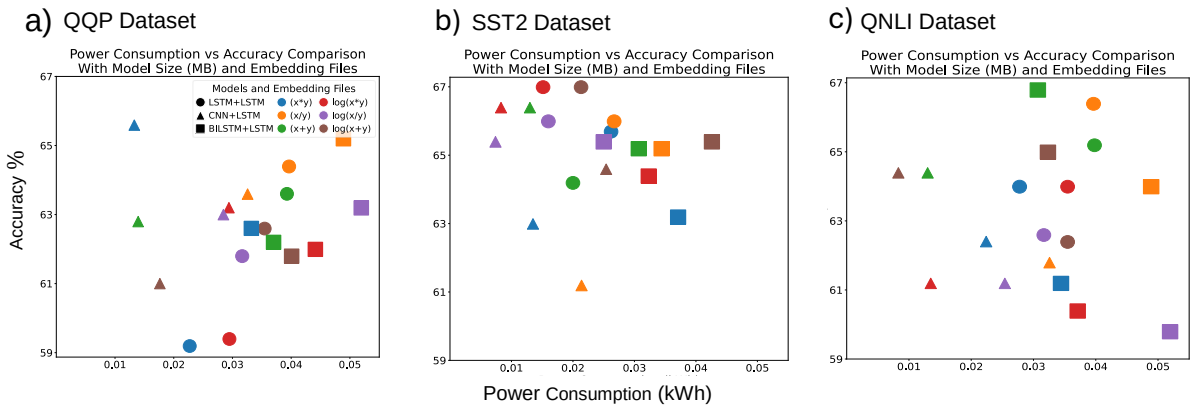


Figure 11: Comparison of Power Consumption and Accuracy with Model Size (MB) and Embedding Files a) for QQP Dataset, b) for QQP Dataset, and c) for QQP Dataset. The smaller the shape size, the smaller the model size is. There is a visible trade-off between the power consumption and accuracy of the three CNN/RNN models. If the considered trade-off point is less than 0.03 kWh and the accuracy is between 63 % and 65 %, then the best embedding file is from Figure A (log(x/y)), Figure B (log(x+y)), or Figure C (log(x+y). This trade-off point is based on the' most common balanced point of these figures, so that it can be used as a uniform trade-off point for all three datasets. Although it is evident that the BiLSTM+LSTM model achieves higher accuracy than other models, it also consumes the highest power. On the contrary, the LSTM+LSTM and the CNN+LSTM have better accuracy and less power consumption, which is very effective for low-end devices. Finally, the CNN+LSTM model outperforms the other two models in terms of accuracy, power consumption, and model size, as its trained model size is smaller. It also uses less electricity and offers improved accuracy.

A.22 Experiment on Discrete Fourier Transform following PCA and SORT in reduced dimensions

		50		75		100		150	
Datasets	Models	PCA	SORT	PCA	SORT	PCA	SORT	PCA	SORT
QNLI	CNN+LSTM	63.59	60.79	62.99	60.59	61.59	61.59	63.39	61.99
	BiLSTM+LSTM	61.99	60.19	62.19	59.39	62.19	60.19	61.99	60.99
	LSTM+LSTM	65.59	59.59	65.19	56.59	64.79	61.00	64.99	60.59
QQP	CNN+LSTM	58.99	62.39	66.60	64.00	64.19	62.20	65.19	63.60
	BiLSTM+LSTM	62.79	62.60	64.59	63.79	64.99	64.00	64.19	62.39
	LSTM+LSTM	61.20	61.80	63.80	62.00	60.80	63.19	63.99	62.00
SST2	CNN+LSTM	62.79	65.79	64.79	65.79	65.59	66.79	64.19	65.39
	BiLSTM+LSTM	63.79	66.59	65.99	65.79	65.79	65.99	67.99	65.99
	LSTM+LSTM	69.99	68.59	67.39	67.39	68.99	66.79	69.39	67.60
CoLA	CNN+LSTM	64.79	58.99	63.79	50.00	63.99	63.39	64.19	62.19
	BiLSTM+LSTM	67.80	60.20	61.00	62.00	62.60	61.59	61.80	64.59
	LSTM+LSTM	62.00	58.79	63.59	56.79	64.39	59.19	62.20	58.00

Table 21: Comparing the reduction methods used in the spectral analysis of contextual embeddings. PCA represents the selection of the desired dimension (n), as in Algorithm 1, done by taking the principal component, and SORT stands for the same, which has been done by sorting the top-n spectrum from V_k^T. The PCA method outperforms the SORT approach for n = 50, 75, 100, and 150 across the three stacked CNN/RNN models.