Strength lies in both: a blend of static and contextual word-information improves performance of low-resource NLP

Anonymous ACL submission

Abstract

Low-resource NLP often suffers because of insufficient computing resources and data scarcity. Specifically, low-resource autonomous devices and resource-constrained environments require a low memory footprint, optimal accuracy for scarce data resources, and reproducibility of the results. To address these issues, we combine contextual and static information of a word to form a blended embedding. Blended embedding and CNN/RNN fusion models optimize against energy cost, inference time, and carbon emission, maximizing the NLP accuracies while avoiding resource-intensive transformer models such as the BERT and its low-resource variants. Experimentation with a few GLUE datasets demonstrates that the developed models compete with other low-resource solutions, such as the DistilBERT, mBERT, TinyBERT, and BERT-mini, with the advantage of higher accuracy and low energy cost. In addition, blended embedding exhibits the potential to achieve better reproducibility of model performance, measured by a reduction of the standard deviation of NLP accuracy. Besides, the cartography analysis done on training samples shows that blended embedding reduces hard-to-learn data. The proposed work provides a viable solution for NLP applications in resource-constrained environments, such as mobile devices and other embedded platforms.

1 Introduction

A trade-off between the energy cost of the NLP model training and the accuracy is essential in the development of NLP models in resource-constrained environments (Han et al., 2015; Strubell et al., 2019; Lin et al., 2020), such as voice-controlled task completion in home automation and healthcare. Maximizing the accuracy of NLP necessitates consideration of memory requirements and energy consumption, which are both vital factors in this domain. In contrast, with the advent of transformer models, most State-of-the-Art (SOTA) NLP models rely on transformer-based implementation in their original or miniature forms, using the contextual information of words in contrast to static information. However, the SOTA models are a few hundred megabytes to gigabytes (Devlin et al., 2018; Brown et al., 2020) in size, making the models impertinent for memory and energy-constrained applications. In addition, the SOTA models require extensive training data, which is often infeasible for languages with insufficient speakers or online resources (Joshi et al., 2020).

In earlier works, quantization (Lam, 2018) and other algorithmic adjustments (Ling et al., 2016; Kim et al., 2020) were pivotal in reducing the memory footprint of word embedding. For instance, authors in (Shu and Nakayama, 2017; Kim et al., 2020) minimize the number of parameters for word representation by representing words as discrete codes. While these methods focused more on how optimally the embedding vectors can be compressed, reduction of the dimensionality of the available embedding vectors, for instance, GloVe (Pennington et al., 2014), word2vec (Mikolov et al., 2013), etc., have not been studied. However, other works have considered dimensionality reduction through post-processing (PPA), the publicly available static word embedding. One such seminal work (Mu et al., 2017) subtracted the mean embedding vector μ of the vocabulary and formed an isotropic embedding by removing the projections of the directionality identified using Principal Component Analysis (PCA). An immediate 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 to the dimensionality reduction of the embedding vector dimension. However, these PPA-PCA approaches enhance the performance of static embed-

ding only (Pennington et al., 2014; Mikolov et al., 2013), most SOTA NLP models rely on contextual word information (Devlin et al., 2018).

These models discussed exhibit substantial environmental implications and incur additional financial expenditures. Namely, energy production relates to CO_2 emission and the resultant heat released into the environment, leading to a complex issue (Strubell et al., 2019). The deployment of SOTA NLP models presents considerable challenges, requiring the careful navigation of the trade-off between accuracy and energy expenditure (Cai et al., 2020). To outwit the problems, this paper introduces a blended word embedding by combining post-processed static embedding and spectrally reduced dimension contextual information. Specifically, the blended embedding harnesses the static resources such as GloVe and word2vec, following a series of post-processing steps performed by using the specific post-processing algorithm (PPA) (Mu et al., 2017) and its extension (Raunak et al., 2019). Analogous to frequency variation of contextual information through the Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), and Gaussian High Pass (GHP) filter. We process contextual information of words and merge it with the processed static information.

We used spectral analysis, high pass filtering, sub-word, and modified contextual embedding to reduce the embedding dimension. Upon spectral filtering, the contextual information is merged with static information to obtain a shorter blended embedding yet to achieve classification accuracy comparable to more extensive static or contextual word embedding. Another caveat in deep learning models is the reproducibility of the results (D'Amour et al., 2022; Summers and Dinneen, 2021), even in identical training performed using the same data. The proposed blended embedding is potentially feasible to explore, showing more excellent reproducibility across different datasets and candidate models. To investigate how a blended embedding offers improved reproducibility, we generated dataset cartography (Swayamdipta et al., 2020) of the training data, which hints at a reduction of hard-to-learn data statistics for the blended embedding. Also, as an alternative to the memory-intensive SOTA models, we identify a bi-layer CNN/RNN core following similar models considered earlier (Maheen et al., 2022) but optimize against accuracy, energy cost, carbon emission, and inference time. Because these CNN/RNN models are of reduced size and are prunable, they may be viable alternatives to eschew the traditional BERT-based implementation for low-resource environments, such as the DistilBERT, TinyBERT, mBERT, and BERT-mini, which often have higher memory footprints on resource-constrained devices. The specific contributions made in this study are as follows:

- Developed some spectral-based dimensionality reduction methods of contextual information and demonstrated potential benefits using data cartography and reproducibility.
- Identified optimal mathematical functions to blend static and contextual information.
- Identified a bi-layer deep neural network (DNN) which, together with the blended embedding, outperforms miniature BERT models in low-resource NLP.

2 Fusion of Static and Contextual Embedding

Let us consider that vector $\nu^s = \{s_1, s_2, \ldots, s_M\}$ and $\nu^c = \{c_1, c_2, \ldots c_N\}$ are the static (S) and contextual (C) embedding, respectively, for word w. To fuse v^s and v^c as inputs requires the same dimension ($M = N$), which is achievable through dimensionality reduction methods such as the PCA, an Autoencoder (Hinton and Salakhutdinov, 2006), or spectral analysis. The contextual embedding, denoted as $\nu^c(w)$ of dimension 768, is extracted using a Global Max Pooling over all the BERT layers. Subsequently, we perform a spectral analysis using the DFT and DCT of the contextual vector v^c, and the DFT maximizes accuracy as an average over different models and GLUE datasets (see Appendix A.11). The DFT analysis of the real-valued contextual embedding vector v^c represents the frequency component in the vector. The frequency-selective filtering reduces the DFT (ν^c) vector of size N to a reduced dimension $(\hat{\nu}^c)$ (Fig. 1b). The reduced representation $(\hat{\nu}^c)$ can be decimated further using the PCA method. We extract the processed contextual information of the given word w using the indexes obtained in the PCA process on the reduced DFT representation of a word w, denoted as $\tilde{\nu}^c$. Finally, a collection of mathematical functions $G(\tilde{\nu}^s, \tilde{\nu}^c)$, as in Eq. 4, optimize the fusion of the merged static and contextual information and generate the blended embedding of a word w.

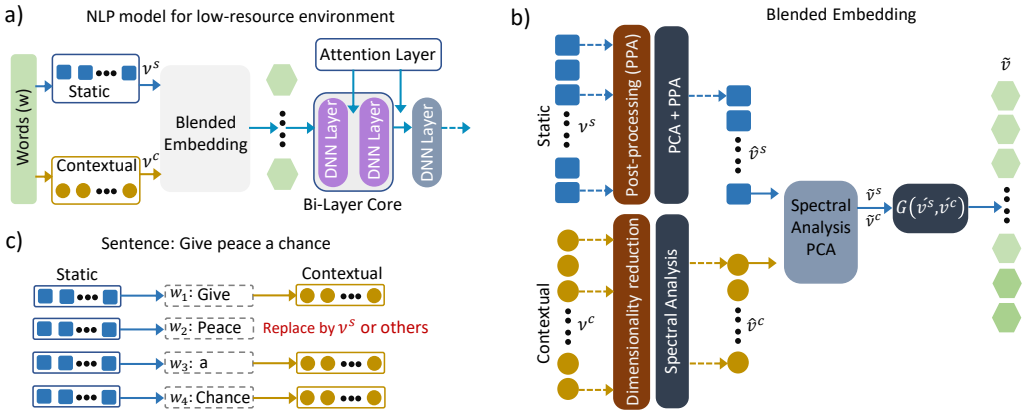

Figure 1: a) Proposed Blended embedding model for low-resource NLP: Both static and contextual embeddings are fused to produce a blended embedding. A bi-layer core with an optional third layer extracts features. b) The blended embedding treats static (through PPA + PCA) and contextual information (through DFT) to produce modified embedding $\widehat{v}^S$ and $\widehat{v}^C$, respectively. With an optional spectral analysis step, a merger function produces the blended embedding $\tilde{v}$. c) In low-resource cases, if the contextual embedding of words are not found in the pre-trained model, the corresponding static or other values fill in.

2.1 Post-processing of the static information

Static embeddings undergo recursive processing upon extraction from pre-trained contextual models such as GloVe and word2vec. We consider the PPA (Mu et al., 2017) and its extension consisting of PCA and an additional PPA (Raunak et al., 2019) as the core to design the modified PPA implemented in Fig. 1b. Here, the recursive calling of PPA+PCA core and subsequent accuracy comparison selects the best-performing static embedding vector form among the competitive options listed as

- PPA as in (Mu et al., 2017), and PCA only
- PPA + PCA as in (Raunak et al., 2019)
- PPA + PCA + PPA + PCA, termed as recursive

Precisely, among the various data sets (QNLI, QQP, SST-2, CoLA, SNLI), considered for comparison of the structure of the PPA + PCA approach (Raunak et al., 2019) performs better (see Appendix A.1). For the static part in the blended embedding, we used GloVe (Pennington et al., 2014). Only using GloVe will reduce the computational cost for the subsequent procedures.

2.2 Spectral analysis of contextual information

The contextual information for each word w_i in a sentence $S = \{w_1, w_2, \ldots, w_n\}$ is from pre-trained BERT layers. After extracting w_i from each BERT layer, a Global Max Pooling produces the contextual vector v^C of the sentence S of interest. We perform the Discrete Fourier Transform (DFT) and Discrete Cosine Transform (DCT) on $\nu^c = \{c_0, c_1, \ldots c_{N-1}\}$ as

$$DFT(V_k) = \sum_{n=0}^{N-1} v_n e^{-j(2\pi/N)nk} \quad (1)$$

$$DCT(V_k) = \sum_{n=0}^{N-1} v_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right], \quad 0 \le k \le N-1 \quad (2)$$

where $V_k = \{V_0, V_1, \ldots V_{N-1}\}$. For each entry n in a word vector $v(w)$ of dimension N, DFT generates V_k, the Fourier spectrum at frequency element k. The magnitude $|V_k|$ of the power spectra represents the strength of the sinusoidal component of frequency k. As an initial step, we calculate the mean level of strength (defined as γ) and conditioned on the magnitude of each spectrum $|V_k|$ to be greater than a threshold α. This is analogous to selective filtering and produces a reduced embedding vector $\widehat{v}^c$. The final phase of the dimensionality reduction steps considers both PCA and sorting approaches and compares their performances over a contextual representation using a vector of 50, 100, 75, and 150 dimensions (see in Appendix A.10)

2.3 Energy cost and carbon footprint analysis

In the assessment of carbon emissions and following training times, all the computational resources using the measurement methods outlined in Strubell et al. (2019). Specifically, we used the default settings and sampled GPU and CPU

power consumption during the model training (50 trials for each selection of 500 data points). The model training process uses a single Google Colab GPU (T4) and CPU, with a few exceptions where a Google Colab P100 GPU is used instead of the T4 and CPU. We computed the total power usage in kilowatt-hours (kWh) using the training times as follows. Let P_c represent the mean power draw throughout the 50 trials. For the power draw quantity, access to the GPU's statistics is necessary, and it allows us to monitor the GPU's power consumption P_g during training. The total power drawn from memory (RAM) during training has been used to determine P_r. The total power P_t usage is the sum of the GPU, CPU, and RAM power consumption and is as follows

$$P_t = 1.58t(P_c + P_g + P_r)/1000 \quad (3)$$

The CO_2 emission calculation from the total power P_t as in follows Eq. 3 the formulation $CO_2e = 0.954P_t$ along with the other details and assessments as summarized in Strubell et al. (2019).

3 Experimental Setup

3.1 Optimal CNN/RNN stack model search

We performed an exhaustive screen of alternative fusion combinations of CNN and RNN modules of length ≤ 3. The fusion model stacks the CNN and RNN layers and adds a self-attention layer (Vaswani et al., 2017), generating 105 different models to study. The initial screen uses a mean accuracy score of over three (QNLI, QQP, SST-2) of GLUE benchmark datasets (Wang et al., 2018) and chooses 19 preliminary models (Appendix A.9). These models were later explored further, considering energy requirement, carbon footprint, inference time, and accuracy to identify the optimal CNN/RNN models.

3.2 Merging functions for static and contextual information

The blended embedding includes static and contextual information, schematically and specifically shown in Fig. 1, a mathematical function stitches them together. A crucial step in blended embedding is optimally fusing different word representations while maximizing classification accuracy. We tried both linear and non-linear strategies and screened the options for a few selected models and QQP, QNLI, CoLA, SST2, and SNLI datasets. The exact merging options are $G(\tilde{\nu}^s, \tilde{\nu}^c)$, considered here are

$$\begin{aligned}
&\tilde{\nu} = \log(\tilde{\nu}^s + \tilde{\nu}^c),\ \tilde{\nu} = \tilde{\nu}^s + \tilde{\nu}^c,\\
&\tilde{\nu} = \log(\tilde{\nu}^s \odot \tilde{\nu}^c),\ \tilde{\nu} = \tilde{\nu}^s \odot \tilde{\nu}^c,\\
&\tilde{\nu} = \log(\tilde{\nu}^s / \tilde{\nu}^c),\ \tilde{\nu} = \tilde{\nu}^s / \tilde{\nu}^c,\\
&\tilde{\nu} = (\alpha \cdot \tilde{\nu}^s) + ((1-\alpha) \cdot \tilde{\nu}^c),\\
&\tilde{\nu} = \sqrt{(\tilde{\nu}^{c^2} + \tilde{\nu}^{s^2} + 2 \cdot \tilde{\nu}^c \cdot \tilde{\nu}^s \cdot \cos(\theta))},\\
&\tilde{\nu} = \text{Concat}(\tilde{\nu}^s, \tilde{\nu}^c),\ \tilde{\nu} = \text{Concat}(\tilde{\nu}^c, \tilde{\nu}^s)\\
&\text{Here, } \alpha = 0.3, 0.5, 0.7; \theta = 0, 30, 45, 60, 90 \quad (4)
\end{aligned}$$

Algorithm 1 Spectral analysis of 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) as in Eq. 1,
Calculate μ = mean($|V_k|$)
Identify indexes in $|V_k|$, where $|V_k| \geq 0.1\mu, \forall k \in \{1, 2, \ldots N-1\}$, or Gaussian High pass filter (GHP)
Use these indexes to produce a reduced DFT or DCT spectrum, V_k^r
Take n-point PCA (V_k^r, n)
Extract the n-indexes of the original DFT spectrum using PCA outcomes
Use these n-indexes to retrieve $\tilde{\nu}^c$ from the original $\tilde{\nu}^c$.

Models	QQP	QNLI	SST2	CoLA	SNLI
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
CNN+LSTM	62.00	**64.99**	**74.79**	**63.00**	47.77
LSTM+LSTM	60.00	**64.99**	73.99	59.20	50.66
BiLSTM+LSTM	**66.50**	63.00	72.19	60.80	49.11

Table 1: This report provides a comparative analysis of performance metrics, averaging five accuracy values across various models and datasets. The study utilizes 500 randomly selected data points, with an additional focus on 1,800 data points from the SNLI dataset. Accuracy values are based on the maximum results of 50 trials on the same data points. For CNN and RNN models, GloVe static embeddings with a 300-dimensional representation are employed.

4 Results

4.1 Bi-Layer CNN/RNN cores are pertinent for low-resource environment

Amongst many, one of the strategies we considered to deal with deployment problems of NLP models in resource-constrained environments would be minimizing the model size while maintaining a competitive accuracy. As we identify here,

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
STDEV	**0.78**	0.84	1.48	1.26	1.00	1.72

Table 2: 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 Discrete Fourier Transform (DFT), and the SORT stands for the same, has been done by sorting top-n spectrum from V_k^r. Additionally, spectral analysis of contextual embeddings by using the Discrete Cosine Transform (DCT), Gaussian High-Pass (GHP)-that allows only the high-frequency spectrum, One hot sentence encoding with DFT (Sent. DFT), and the embedding from BERT-mini following the DFT algorithm are shown. The DFT with PCA method outperforms all the other approaches.

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 3: 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 mergers, with an average of 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.

stacking CNN/RNN layers provides leverage over widely practiced BERT-based miniature models for several textual classification tasks included in the GLUE benchmark datasets. Earlier works provide evidence of the superior performance of RNN and DNN fusion extending beyond the earlier approaches. Precisely, from the extensive search of the competing models comprising stacking CNN/RNN layers, a few models emerge that outperform some low-resource BERT-based models based on model size and energy cost, as summarized in Appendix A.9. Also, these models have low energy costs and carbon footprint over the other alternative models and the BERT-based miniature versions in Table 1.

4.2 Optimal Method search to combine static and contextual information

An important question associated with blending embedding is the mechanism to combine static and contextual information. Against two objectives- i) less power consumption and ii) higher accuracy- we assess a few mathematical functions as in Eq. 4 over multiple datasets and models (see Appendix A.8, A.11). As observed, there is a visible trade-off between the power consumption and accuracy of the three CNN/RNN models' alternative merging functions studied. For instance, if the considered trade-off points are less than 0.01 kWh and the accuracy is between 63 to 68 percent for the $CNN + LSTM$ model, we identify blending functions $\log(x/y)$ and $x + y$ in Fig. 4. Considering the performance in other models, the functions $\log(x/y)$, $(x + y)$, and $\log(x + y)$ are better-

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_SC	44.22	47.11	48.66
Concat_CS	44.66	47.33	47.33

Table 4: This comparison shows the performance across different blending methods with 75 D over the SNLI dataset with 1800 training samples. Overall performance for the alpha blend for 0.5 is giving better accuracy following the formulation from Eq. 4.

Models	Using	Mean Accuracy	Standard Deviation
LSTM+ LSTM	GloVe	59.85	**1.71**
	Fourier	**60.53**	1.85
	Max Pooling	59.72	2.48
	PCA		
CNN+ LSTM	GloVe	**61.98**	**0.87**
	Fourier	59.89	1.68
	Max Pooling	61.47	2.03
	PCA		
BiLSTM+ LSTM	GloVe	59.15	1.76
	Fourier	60.78	**1.40**
	Max Pooling	**61.05**	1.87
	PCA		
RoBERTa		55.94	3.29
BERT-mini		55.64	3.41
DistilBERT		57.39	4.49
TinyBERT		51.09	4.56

Table 5: The models' reproducibility was evaluated with and without applying a Fourier transformation to the embedding file. This evaluation compared GloVe and various BERT variants, highlighting the differences in mean accuracy, variance, and standard deviation. The average of the QQP, QNLI, CoLA, SST2, and SNLI datasets was taken for Comparison.

performing blending functions. Besides, among the competing fusion models, the overall performance of the CNN+LSTM model appears better than the other two (LSTM+LSTM, BiLSTM+LSTM) models in power consumption and size. As obtained here, the improvement achieved by blended embedding depends on the mathematical functions, for instance function $\log(x/y)$ and $x + y$, outperform other alternatives functions and strategies studied (see Table 4).

4.3 Spectral analysis concisely represents contextual information

Upon extracting the contextual information, the DFT analysis on $\tilde{\nu}^c$ transforms the numeric sequence into the spectra of different sinusoidal frequencies. The magnitude of each spectrum proportionately represents the strength of the corresponding frequency. Instead of removing high or low-frequency components, we emphasized decimating the frequencies of the weakest strength, mimicking selective filtering of the Fourier spectra. The spectral analysis keeps widely varying bands of frequencies. It may relate to contextual information variation over different scales, often seen from sentence to document level in many NLP contexts (Tamkin et al., 2020). As observed in Table 6, the blended embedding of size 75 achieves higher accuracy frequently over five alternative datasets and model choices. However, the blended and pure contextual embedding of the size 75 vector performs inferiorly to GloVe. One plausible explanation could be that SST2 samples chosen randomly need little or no contextual information during classification. We found that the DFT-based spectral analysis increases the cosine similarity distance between the reference embedding (BERT-mini) and the embedding extracted by DFT, prohibiting cluster formation of points on the plane of cosine similarity analysis (Fig. 3). One potential contribution of such dispersed dissimilarity map of DFT-based reduced contextual embedding is to enhance the reproducibility of the classification accuracy, which is pursued further.

4.4 Blended static and contextual information improves NLP performance

Among the merging functions and their reproducibility explored in Eq. 4, here we use a simple point-wise addition of the modified static and contextual information to improve the classification accuracy. As shown by the accuracy comparison of QNLI, QQP, SST2, CoLA, and SNLI in Fig. 1 for the two stacked CNN/RNN models obtained through model search (shown in Table 3). Also, the blended embedding improves classification accuracy compared to cases where the models were trained only using the static embeddings, which is evident from the data in Appendix A.5, and detail experiment is in Appendix Tables 11, 12, 13 for three datasets. Precisely, the performance of each model improves by a margin of 1-2% or even

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	56.05	61.12	55.64
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	56.05	61.12	55.64

Table 6: Comparison of the accuracy between the proposed blended embedding and other off-the-shelf approaches. Here, the average of the QQP, QNLI, SST2, CoLA, and SNLI are calculated for each model. The blended embedding achieves a competitive classification score even with a comparatively smaller embedding vector dimension when compared to GloVe 300. The blended approach demonstrates a comparable match with contextual-only embedding.

more when the blended embedding (of size 75) is used instead of the static embedding (of size GloVe-300) only. Such reduction of embedding dimension without compromising accuracy is advantageous from the memory footprint and energy cost perspective; low-resource NLP is of immense interest. Compared to miniature BERT models, the fusion models consistently demonstrate superior performance across nearly all instances examined in this study.

In addition, using spectral analysis in the blended embedding improves the reproducibility of NLP models' accuracy, contributing a way forward for the non-determinism of the deep learning models. The comparison done over QQP, QNLI, CoLA, SNLI, and SST2 datasets for the three identical CNN/RNN fusion models that blended embedding reduces the standard deviation (σ) of the classification accuracy calculated for 20 different randomly chosen datasets of size 500 and 1800 for SNLI dataset only. Here, reducing in σ of accuracy represents better reproducibility. As, it has been studied over multiple datasets and models, the blended embedding produced mainly by the DFT-PCA approach reproduces a reduced accuracy compared to approaches devoid of DFT-based spectral analysis. Precisely, the proposed DFT-PCA embedding achieves a minor standard deviation in most of the combinations considered and surpasses models such as DistilBERT, mBERT, BERT-mini, and TinyBERT by a considerable margin. How such a reduction in σ is achievable and how the blended embedding appears superior are questions we investigate further using data cartography as in Fig. 5 (see Fig. 2, and Appendix A.12, A.16).

How much power an NLP model harnesses has been crucial for applications with scarce computational resources. Specifically, applications such as those that require edge devices often rely on extensive energy budgeting of the computing devices, thereby requiring NLP models to harness the least energy during the classification task. The proposed blended embedding may be a viable avenue to navigate further for such energy-constraint NLP applications. For instance, a comparison between GloVe and blended embedding, as in Appendix A.4, shows that blended embedding mostly outperforms GloVe accuracy while consuming less energy from the source. Besides, the model options are all compared against two performing objects focused on low-resource NLP in Pareto-front-like analysis (see Fig. 4) to select model options and merger methods for static and contextual information. Such exploration protocol addresses low-resource NLP, and along with blended embedding achieving higher accuracy (see Table 6, and Appendix A.5) make our work a viable alternative for low-resource NLP.

5 Discussion

Semantic embedding transforms words into real numbers— one of the subclasses is the static embedding approach that uses the probabilistic appearance of a word in a large corpus as its central dogma to generate a vector representation of a word. In contrast, the contextual embedding encapsulates the underlying context of word usage in a sentence. While both approaches have pros and cons, contextual embedding largely produces state-of-the-art textual classification accuracy and is widely used. We combined these approaches with custom mathematical transformations to form the blended embedding, demonstrating enhanced classification in our preliminary analysis. Besides, devices that lack sufficient memory storage and processing power often fail to harness SOTA models and may be equipped with stacked CNN/RNN models identified through extensive screening of alternative design choices. Also, 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-

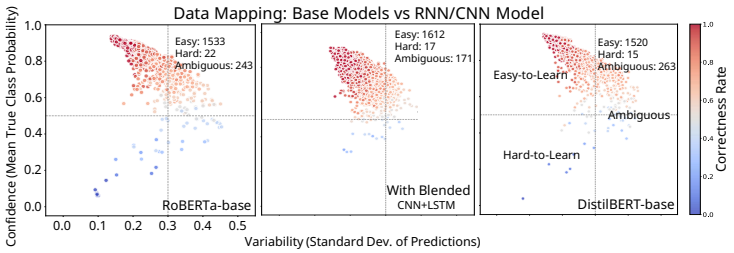

Figure 2: A data mapping comparison among RoBERTa-base, DistilBERT-base, and CNN+LSTM models for the SNLI dataset. Detailed analyses of data mapping are shown in Fig. 7, and Fig. 8 with $\alpha = 0.5$. Blended embedding performs better in low-resource environments with CNN/RNN models.

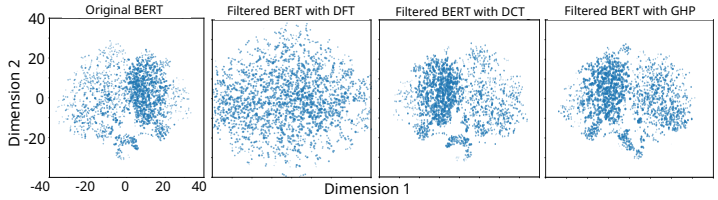

Figure 3: This visual representation shows a comparison among the original BERT-base extracted embedding and the differences after applying the Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), and Gaussian High pass filter (GHP). In DFT processing, it reduces most of the weak frequencies and becomes a tightly clustered embedding, indicating a strong reduction of noise/outlier.

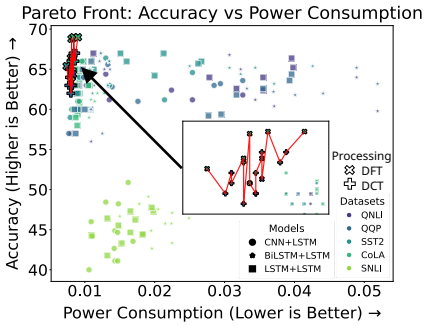

Figure 4: The blended embedding treats static (through PPA + PCA) and contextual information (through DFT and DCT) to produce modified embedding $\widehat{v}^S$ and $\widehat{v}^C$, respectively. With an optional spectral analysis step, a merger function produces the blended embedding $\tilde{v}$. Here, the log(x/y) performs well with the DFT.

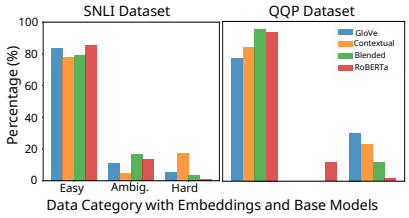

Figure 5: Performance comparison among static, contextual, and blended embeddings is presented for the SNLI and QQP datasets. The average of three CNN/RNN models is taken, and the insight is in blended embedding; the hard-to-learn rate is decreasing, and the easy-to-learn and ambiguous rates are increasing with minimum computational resources, almost close to the RoBERTa and DistilBERT base models.

process the contextual embedding, we hypothesize that alternative forms of filtering would be worth it, considering the frequency dependency of contextual information across various NLP tasks.

Limitations

This study provides a newly devised blended embedding that resorts to static and contextual information to maximize the accuracy of a low-dimension word representation. The investigation could be extensively applied to other GLUE benchmark datasets and low-end devices for further tuning and model size adjustments. Moreover, replacing the missing words with static GloVe word vector gives better performance, as schematically shown in Fig. 1. The identification of an optimal strategy to minimize the occurrence of missing words requires further investigation. Other contexts, such as class imbalance and even smaller datasets, can be tested exhaustively to prove their applicability. Besides, a more acute sense of uniformity of hardware and computing facility while training and running the NLP models is necessary, which is occasionally compromised because of device switching in the Colab environment.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 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*.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. 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*.

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.

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, et al. 2020. Mcunet: Tiny deep learning on iot devices. *Advances in Neural Information Processing Systems*, 33:11711–11722.

Shaoshi Ling, Yangqiu Song, and Dan Roth. 2016. 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.

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*.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017. 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.

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.

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.

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*.

A Appendix

A.1 Recursive Post Processing with GloVe

		100 D		300 D
Data	PPA	$PPA_2 - PCA$	$PPA_3 - PCA_2$	$GloVe$
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 7: 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.
$PPA_2 - PCA : PPA - PCA - PPA$
$PPA_3 - PCA_2 : PPA - PCA - PPA - PCA - PPA$

A.2 Data Mapping Experiments with Several Embeddings

Embeds	Accuracy (%)	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
BERT GHP 300	46.00	26.56	71.33	2.11
log(x+y) 75	48.51	38.89	50.44	10.67
log(x/y) 75	47.62	**50.11**	**32.22**	**17.67**
One hot sent. Embed DFT 300	47.99	**53.56**	**22.78**	23.67
BERT-mini DFT 250	43.33	46.00	40.00	14.00

Table 8: Comparison with the accuracy and the data cartography for different types of embeddings. This experiment used the SNLI dataset with three CNN/RNN models selected: LSTM+LSTM, CNN+LSTM, and BiLSTM+LSTM. It shows the average percentage of the three models where easy-to-learn data samples are increasing with the blended embedding log(x/y), and the dimension n = 75. Also, the hard-to-learn samples decrease with the blended embedding or keep a minimal difference compared to GloVe, BERT DFT, and BERT DCT, where dimension n = 300. Moreover, the percentage of ambiguity also decreases compared to GloVe 300D.

A.3 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 three-layer model

A.4 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.5 Accuracy comparison among the Blended embedding ($\log(x+y)$, $\log(x/y)$) 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 10: 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.

SST2				
75 Dimension	Log(x/y)	Log(x+y)	Contextual	GloVe
CNN+LSTM	**63.19**	62.79	66.6	69.59
BiLSTM+LSTM	65.39	**66.59**	67.39	69.99
LSTM+LSTM	66.6	**68.39**	68.19	71.59
100 Dimension				
CNN+LSTM	65.39	66.99	66.19	71.39
BiLSTM+LSTM	65.39	65.79	64.59	70.19
LSTM+LSTM	65.59	64.59	68.59	70.19
150 Dimension				
CNN+LSTM	60.59	66.59	67.39	70.99
BiLSTM+LSTM	69.95	65.59	66.59	71.19
LSTM+LSTM	66.39	67.39	68.99	71.99
300 Dimension				
CNN+LSTM	65.39	66.19	67.59	74.79
BiLSTM+LSTM	70.19	65.99	65.59	72.19
LSTM+LSTM	65.19	68.2	68.39	73.99

Table 11: Comparison of accuracy between the Blended embedding (log(x+y), log(x/y)) (size 75, 100, 150, and 300) vs. Contextual vs. Static embedding (GloVe-300) files. The blended embedding can achieve a competitive classification score even with a comparatively smaller embedding vector dimension. Here, the analysis is done only for the SST2 dataset.

QNLI				
75 Dimension	Log(x/y)	Log(x+y)	Contextual	GloVe
CNN+LSTM	**69.99**	62.99	62.99	66.00
BiLSTM+LSTM	63.99	**68.00**	62.19	62.99
LSTM+LSTM	62.99	**64.99**	65.19	64.99
100 Dimension				
CNN+LSTM	61.99	62.39	61.59	64.00
BiLSTM+LSTM	59.79	64.99	62.19	62.99
LSTM+LSTM	62.59	62.39	64.79	63.99
150 Dimension				
CNN+LSTM	58.59	64.59	63.39	64.99
BiLSTM+LSTM	57.79	63.99	61.99	63.99
LSTM+LSTM	57.79	65.99	64.99	62.00
300 Dimension				
CNN+LSTM	61.40	66.60	58.59	64.99
BiLSTM+LSTM	64.39	63.59	58.39	63.00
LSTM+LSTM	64.79	63.39	56.99	64.99

Table 12: Comparison of accuracy between the Blended embedding (log(x+y), log(x/y)) (size 75, 100, 150, and 300) vs. Contextual vs. Static embedding (GloVe-300) files. The blended embedding can achieve a competitive classification score even with a comparatively smaller embedding vector dimension. Here, the analysis is done only for the QNLI dataset.

QQP				
75 Dimension	Log(x/y)	Log(x+y)	Contextual	GloVe
CNN+LSTM	62.99	**64.99**	66.59	62.00
BiLSTM+LSTM	62.00	**64.99**	64.59	62.99
LSTM+LSTM	61.00	**62.99**	63.79	62.00
100 Dimension				
CNN+LSTM	62.99	61.00	64.19	62.00
BiLSTM+LSTM	63.19	61.79	64.99	62.00
LSTM+LSTM	61.79	62.59	60.79	63.99
150 Dimension				
CNN+LSTM	63.19	62.39	65.19	62.99
BiLSTM+LSTM	61.59	63.19	64.19	62.99
LSTM+LSTM	59.19	61.79	63.99	61.00
300 Dimension				
CNN+LSTM	63.59	60.19	65.19	62.00
BiLSTM+LSTM	66.39	61.00	62.39	66.50
LSTM+LSTM	62.59	62.59	64.59	60.00

Table 13: Comparison of accuracy between the Blended embedding (log(x+y), log(x/y)) (size 75, 100, 150, and 300) vs. Contextual vs. Static embedding (GloVe-300) files. The blended embedding can achieve a competitive classification score even with a comparatively smaller embedding vector dimension. Here, the analysis is done only for the QQP datasets.

A.6 Comparison of Missing Words alternative with Contextual, GloVe, sent2vec, and Random initial Methods

				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.7 sent2vec analysis and comparison

Data	**Models**	Accuracy 75	Accuracy 300	Accuracy 700	Power Consumption (kWh) 75	Power Consumption (kWh) 300	Power Consumption (kWh) 700
QQP	CNN+LSTM	62.00	**67.00**	62.00	0.0118	**0.0112**	0.0194
	BiLSTM+LSTM	58.99	**64.99**	62.00	**0.0146**	0.0161	0.0268
	LSTM+LSTM	50.99	**57.99**	54.00	0.0104	**0.0101**	0.0273
CoLA	CNN+LSTM	66.00	**67.00**	56.00	**0.0205**	0.0257	0.0217
	BiLSTM+LSTM	**67.00**	**62.00**	**56.99**	0.0146	0.0238	**0.0126**
	LSTM+LSTM	**57.99**	**63.99**	56.00	0.0105	**0.0103**	0.0109
SNLI	CNN+LSTM	48.88	52.44	**54.44**	**0.0254**	0.0307	0.0467
	BiLSTM+LSTM	50.22	50.88	**54.00**	**0.0398**	0.0451	0.0659
	LSTM+LSTM	48.22	51.99	**55.77**	**0.0452**	0.0521	0.0579

Table 15: Comparison of different dimensions with sent2vec static embedding where dimension n = 75, 300, and 700. Overall performance increases for 300 D compared to GloVe static with 300 D. However, with 75 D, the performance drops significantly.

A.8 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 16: 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 17: 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 with contextual embeddings first (denoted as CS) achieve superior performance with the 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 18: This analysis evaluates two dimensionalities, n = 75 and n = 300, using a cosine formula with different values for θ on the SNLI dataset.

A.9 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 19: 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 SST-2 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 T4 GPU to address potential concerns regarding reproducibility in Table 5. Following the previous studies to identify a fusion length of three as optimal for a DNN stacked structure, we expand the search pool by allowing each layer to host 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.3). Each model was trained for QNLI, SST2, and QQP datasets (Wang et al., 2018) and ranked, considering the mean accuracy of the 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.10 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 20: 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 the SORT stands for the same, has been done by sorting 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.

A.11 Comparison of Static and Contextual Embedding Merging Approaches: DFT, DCT, and 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 21: 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.12 Data Mapping with embeddings Datasets

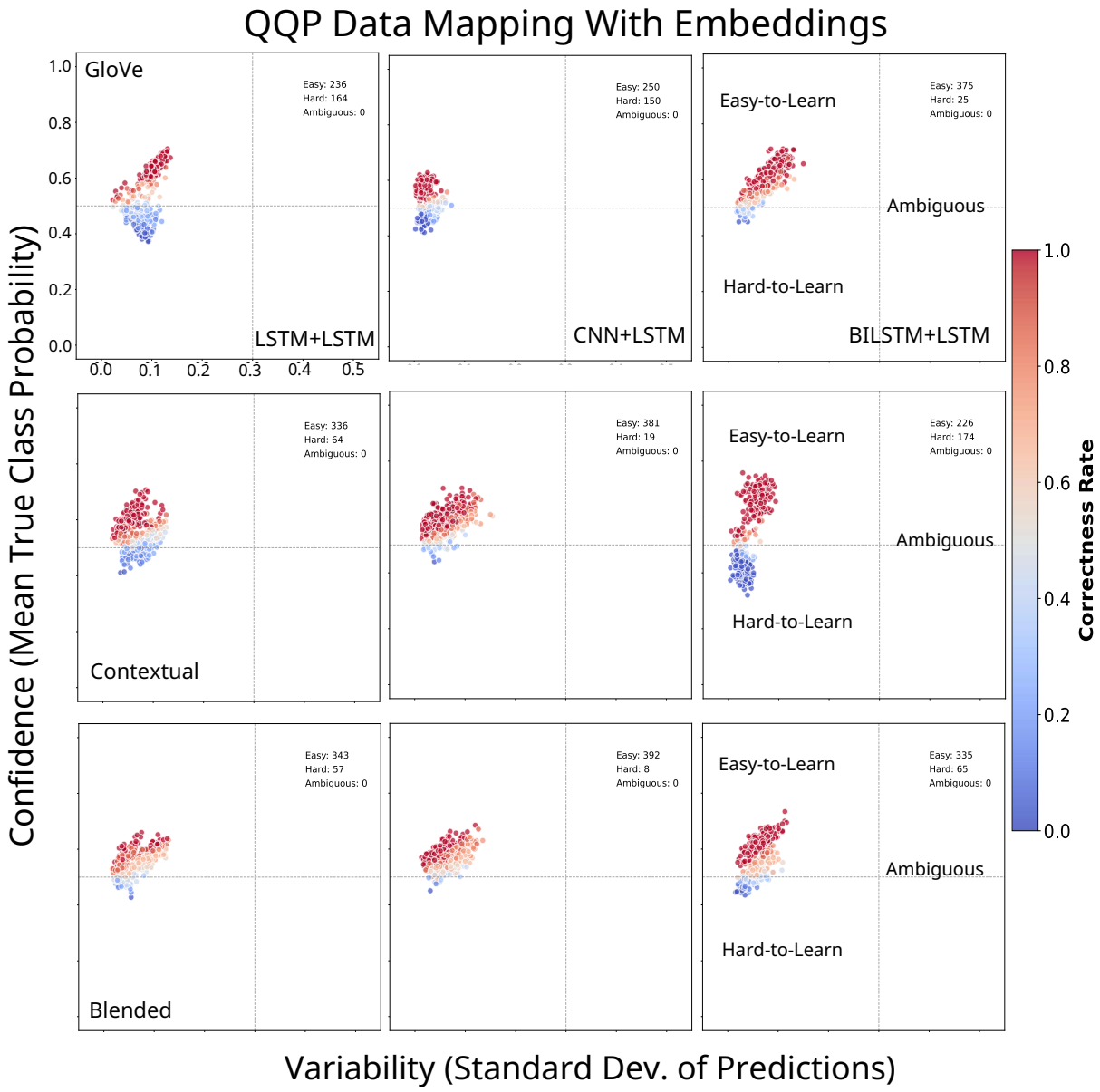

Figure 6: A data mapping comparison among static, contextual, and blended embeddings with three CNN/RNN models for the QQP dataset. In the Blended alpha embedding, the Hard-to-Learn is reduced, and the Easy-to-Learn, Ambiguous is increased. Moreover, the correctness rate of samples is also increased.

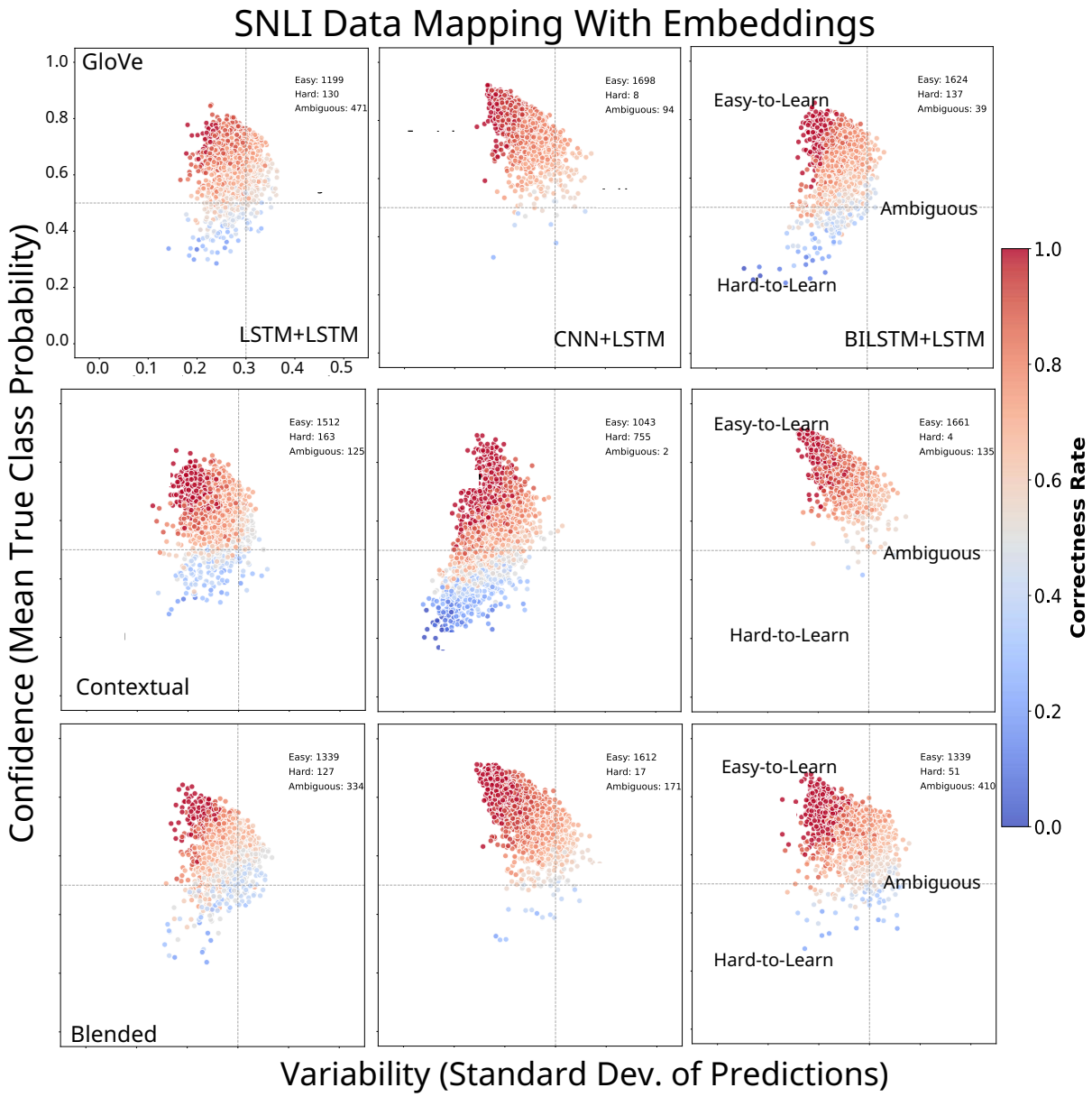

Figure 7: A data mapping comparison among static, contextual, and blended embeddings with three CNN/RNN models for the SNLI dataset. In the Blended alpha embedding, the Hard-to-Learn is reduced, and the Easy-to-Learn, Ambiguous is increased. Moreover, the correctness rate of samples is also increased.

A.13 Data Mapping for Base models

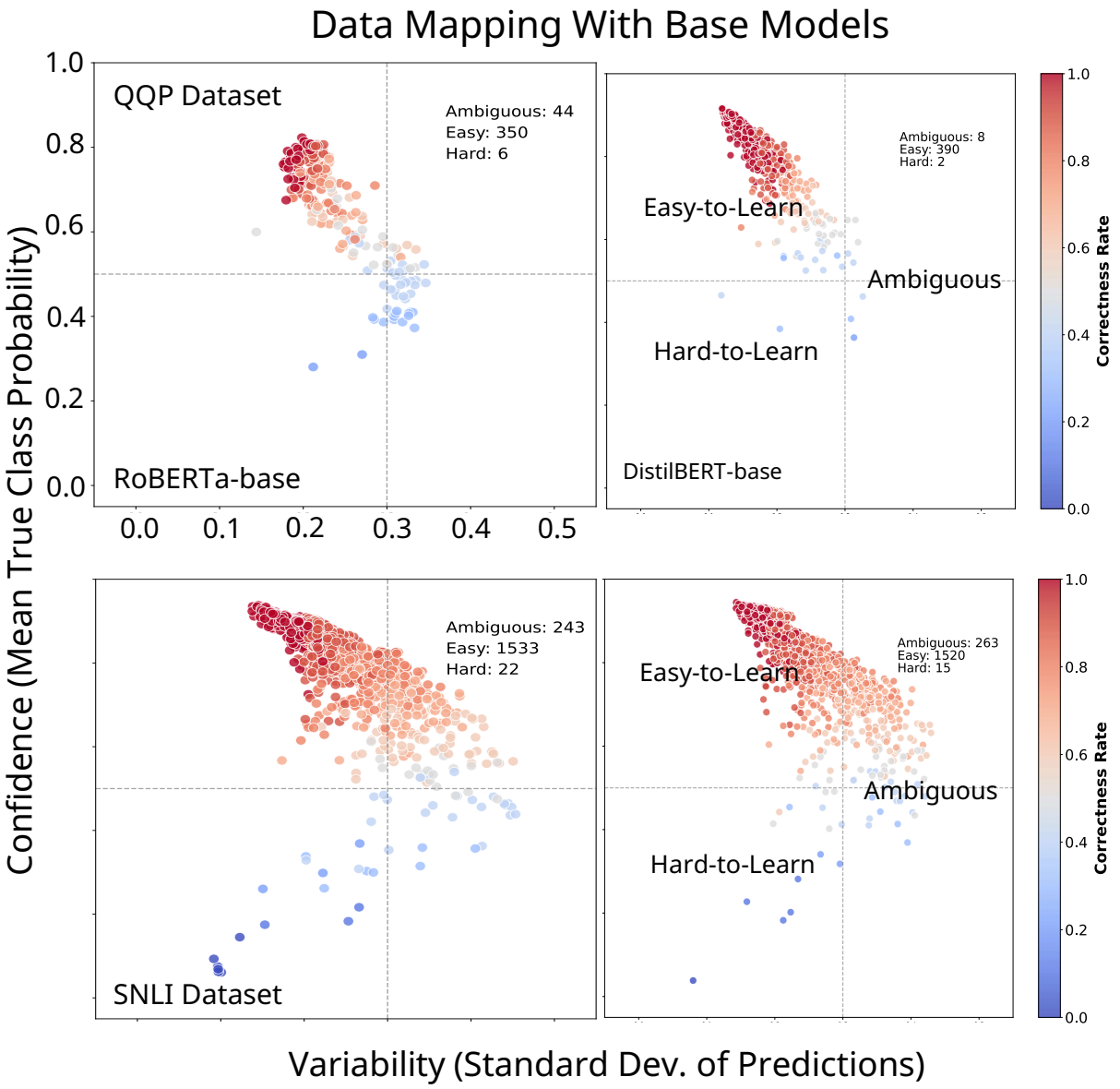

Figure 8: A data mapping of RoBERTa-base and DistilBERT-base models for the QQP and SNLI dataset. Compared to it, QQP and SNLI data mapping, respectively, Fig. 6 and Fig. 7 with the $\alpha = 0.5$ Blended embedding is performing better in low-resource environments with CNN/RNN models.

A.14 Data Mapping with embeddings for BiLSTM+LSTM

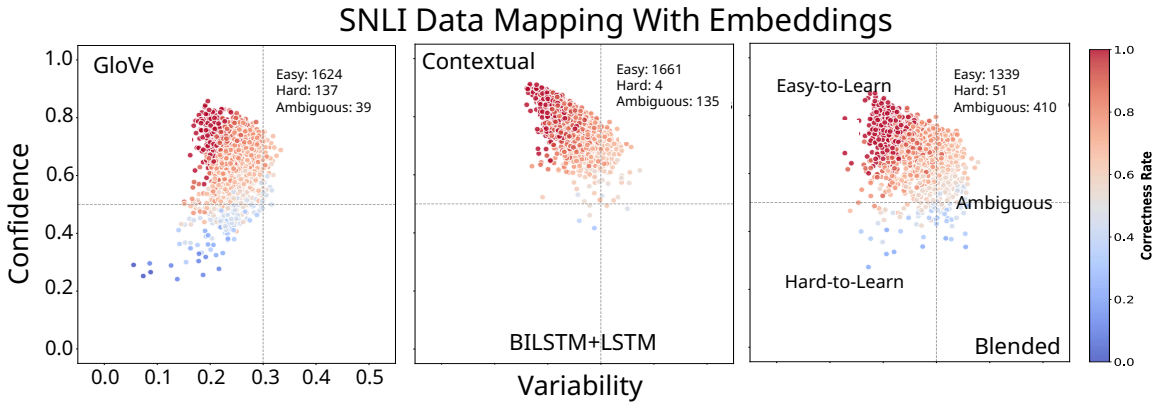

Figure 9: A data mapping of the BiLSTM+LSTM model for the SNLI dataset is performing better compared to other models, static-only and contextual-only, in the aspect of accuracy and correctness. The detailed data mapping for QQP and SNLI datasets are, respectively, Fig. 6 and Fig. 7. Here, the $\alpha = 0.5$ Blended embedding is performing better in low-resource environments with CNN/RNN models.

A.15 Power and Accuracy comparison among the Blended embeddings with their model sizes

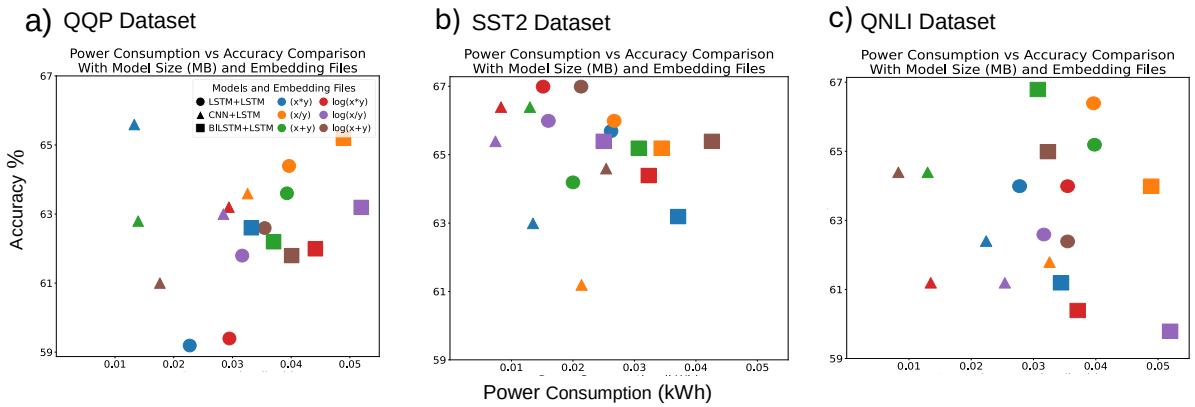

Figure 10: Comparison of Power Consumption and Accuracy with Model Size (MB) and Embedding Files a) for QQP Dataset, b) for QQP Dataset, and c) for QQP Dataset. The smaller the shape size, the smaller the model size is. There is a visible trade-off between the power consumption and accuracy of the three CNN/RNN models. If the considered trade-off point is less than 0.03 kWh and the accuracy is between 63 to 65 percent, then the best embedding file from Figure A, log(x/y), from Figure B, log(x+y), and from Figure C, log(x+y). This trade-off point is based on these figures' most common balanced point so that it can be used as a uniform trade-off point for all three datasets. Though it is visible that the BiLSTM+LSTM model is giving higher accuracy than other models, it is also consuming 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 is better than the other two models in accuracy, power consumption, and model size because its trained model size is less. It also uses less electricity and offers improved accuracy.

A.16 Dataset Cartography Formula

The formula calculates the *confidence* of a model for an instance i as the mean probability assigned to the true label across E epochs.

$$\hat{\mu}_i = \frac{1}{E} \sum_{e=1}^{E} p_{\theta^{(e)}}(y_i^* \mid \mathbf{x}_i) \tag{5}$$

The formula calculates the *variability* of a model for an instance i using the standard deviation of the model's probability for the true label across E epochs.

$$\hat{\sigma}_i = \sqrt{\frac{\sum_{e=1}^{E} \left(p_{\theta^{(e)}}(y_i^* \mid \mathbf{x}_i) - \hat{\mu}_i\right)^2}{E}} \tag{6}$$