Processing Natural Language on Embedded Devices: How Well Do Transformer Models Perform?

Souvika Sarkar*
szs0239@auburn.edu
Auburn University
Alabama, USA

Mohammad Fakhruddin Babar*
m.babar@wsu.edu
Washington State University
Pullman, USA

Md Mahadi Hassan
mzh0167@auburn.edu
Auburn University
Alabama, USA

Monowar Hasan
monowar.hasan@wsu.edu
Washington State University
Pullman, USA

Shubhra Kanti Karmaker Santu
sks0086@auburn.edu
Auburn University
Alabama, USA

ABSTRACT

Transformer-based language models such as BERT and its variants are primarily developed with compute-heavy servers in mind. Despite the great performance of BERT models across various NLP tasks, their large size and numerous parameters pose substantial obstacles to offline computation on embedded systems. Lighter replacements of such language models (e.g., DistilBERT and TinyBERT) often sacrifice accuracy, particularly for complex NLP tasks. Until now, it is still unclear *(a)* whether the state-of-the-art language models, viz., BERT and its variants are deployable on embedded systems with a limited processor, memory, and battery power and *(b)* if they do, what are the "right" set of configurations and parameters to choose for a given NLP task. This paper presents a *performance study of transformer language models* under different hardware configurations and accuracy requirements and derives empirical observations about these resource/accuracy trade-offs. In particular, we study how the most commonly used BERT-based language models (viz., BERT, RoBERTa, DistilBERT, and TinyBERT) perform on embedded systems. We tested them on ***four*** off-the-shelf embedded platforms (Raspberry Pi, Jetson, UP^2, and UDOO) with 2 GB and 4 GB memory (i.e., a total of ***eight*** hardware configurations) and ***four*** datasets (i.e., HuRIC, GoEmotion, CoNLL, WNUT17) running various NLP tasks. Our study finds that executing complex NLP tasks (such as "sentiment" classification) on embedded systems is *feasible* even without any GPUs (e.g., Raspberry Pi with 2 GB of RAM). Our findings can help designers understand the deployability and performance of transformer language models, especially those based on BERT architectures.

---

*Both authors contributed equally to this work.

CCS CONCEPTS

• **Computing methodologies → Natural language processing**;
• **Computer systems organization → Embedded hardware**.

KEYWORDS

Transformers; Embedded Systems; NLP; Language Models

1 INTRODUCTION

The natural language processing (NLP) domain and the emergence of large language models rapidly transform how we interact with technology. With the proliferation of IoT-specific applications, the demand for voice-controlled services that can perform tasks by responding to spoken commands is growing. Use cases of NLP applications, especially those that need embedded and mobile systems, include home automation, healthcare, industrial control, and automotive infotainment. To design a dialogue-based interaction system for a target device, we need models that are feasible to *(a)* run on the hardware and *(b)* meet the desired level of accuracy. Although some services such as digital assistants (e.g., Alexa, Siri, Cortana) may leverage cloud resources for processing human voices, there exist applications (e.g., offline home/industrial robots, automotive infotainment, battlefield/military equipment) that may not have network connectivity, thus require to synthesize NLP tasks on the embedded device itself. The challenge is understanding the *feasibility* of running a large language model on *resource-limited* devices.

Transformer-based architectures [1], especially BERT-based models [2], have established themselves as popular state-of-the-art baselines for many NLP tasks, including *Intent Classification (IC)*, *Sentiment Classification (SC)*, and *Named Entity Recognition (NER)*. However, a well-known criticism of BERT-based architectures is that they are data-hungry and consume a lot of memory and energy; therefore, deploying them in embedded systems is challenging. In fact, due to their excessive size (431 MB) and parameters (110 M), deploying a pre-trained BERT model (called

$BERT_{Base}$) in resource-constrained embedded devices is often impractical, especially at the production level with certain minimum accuracy/performance requirements. Lighter versions of BERT (e.g., DistilBERT [3] and TinyBERT [4]) often result in accuracy losses. The degree of degradation in performance depends on the difficulty of the task, especially since those models often cannot perform well on complex NLP tasks, including emerging entity [5] or mixed emotion detection [6]. Therefore, designers must make an inevitable trade-off between an accurate model and one that can run smoothly in a resource-constrained environment. Unfortunately, developers often have little idea about this trade-off and have to spend a lot of time conducting trial-and-error experiments to find a suitable architecture that is feasible for the target (resource-constrained) hardware and meets a desired level of accuracy.

From a developer's perspective, it is still unclear what is the "right" BERT-based architecture to use for a given NLP task that can strike a suitable *trade-off between the resources available and the minimum accuracy* desired by the user. Due to the staggering size of the $BERT_{Base}$ model, we experiment with different "distilled" BERT models (e.g., DistilBERT and TinyBERT) for IC, SC, and NER tasks. However, existing ready-to-use distilled models perform poorly on some SC and NER datasets (Sec. 4). Hence, there is a need to *explore other models that can better optimize the efficiency/accuracy trade-offs*.

This research performs an *exploratory study of BERT-based models*[1] *under different resource constraints and accuracy budgets to derive empirical data about these resource/accuracy trade-offs*. We aim to answer the following questions: *(a)* how can we determine the suitable BERT architecture that runs on a target hardware and meets user-defined performance requirements (accuracy, inference time)? *(b)* what are the trade-offs between accuracy and model size as we perform optimizations (such as *pruning*) to run them on limited embedded device memory? and *(c)* what are the implications of performing pruning on accuracy and corresponding resource usage, including memory, inference time, and energy consumption? In answer to those questions, we observe the overhead of running various BERT architectures on four different hardware (viz., Raspberry Pi [9], Jetson Nano [10], UP[2] [11], and UDOO [12]). Our experiments suggest that some BERT models (specifically those that are "distilled") failed to achieve desired performance goals (e.g., F1 score) for various NLP tasks. Further, although pruning can reduce model size, it does not significantly help in energy efficiency.

Contributions.   Our study fills the gap between simulation-based studies and real-world scenarios, as no prior work has deployed these models on embedded platforms. The findings of this work can help designers choose alternative BERT-based architectures under given resource constraints, thus saving development time and hassle. To ensure reproducibility, our implementation and related documentation is **publicly available** [13].

We made the following contributions in this paper.

- Our study systematically investigates the performance of BERT-based language models on four off-the-shelf embedded platforms (Raspberry Pi, Jetson, UP[2], and UDOO) with two different memory variants (2 GB and 4 GB RAMs). We analyzed the trade-offs between complexity and accuracy across multiple NLP tasks. (Sec. 3-Sec. 4).
- We explore the feasibility of deploying complex NLP tasks on embedded systems and analyze them under three *metrics*: *(a)* inference time, *(b)* memory usage, and *(c)* energy consumption. We developed a lookup table through empirical observations that will be useful for system designers to decide suitable model configurations for the target platform (Sec. 4).

Our key findings.   The observations from executing the NLP tasks on our test platforms are as follows: *(a)* simpler NLP tasks such as IC can result in a relatively high (90%+) F1 score; *(b)* the time required to perform inference proportional to the size of the trained model and trimming them result in some accuracy loss (e.g., 60% of reduction in model size could reduce the accuracy by 50%); *(c)* the energy consumption on our test hardware remains relatively consistent, whether we prune the model or not; and *(d)* while GPUs play a role in decreasing inference time, the unavailability of GPUs can be compensated by faster CPUs.

We now start with the problem statement and present selected datasets (Sec. 2). Section 3 describes our experiment setup before we discuss our findings in Sec. 4.

2   PROBLEM STATEMENT & DATASETS

We aim to study *how language models can be optimally deployed* to accomplish dialog processing in embedded devices. The core technical challenge of any dialog system is to accurately understand and interpret user "utterances" and perform the right "action" accordingly. At a fundamental level, user utterance understanding relies on the following three basic NLP tasks[2]: *(a) Intent classification (IC)* — to understand the need of the user, *(b) Sentiment Classification (SC)* — to understand user emotions, and *(c) Named-entity Recognition (NER)* — to extract related entities such as persons or objects.

Figure 1 presents the workflow of the dialogue-based systems considered in this work. The user initiates the interaction by providing a spoken command to the voice-controlled device (marker ① in Fig. 1). The device employs existing automatic speech recognition techniques [14, 15] to convert the user's speech into texts as most language models take textual input (②). The system then runs an "intent classifier" (Sec. 2.1.1) and analyzes the extracted text to determine the user's *intention* (③). The classifier identifies the relevant user intentions for the given command (④). For instance, the intents for the given command *"Can you please go to my study room and turn off the lights?"* could be identified as *"Motion"* and *"Change operational state,"* as it instructed the device to move from its current position. Simultaneously, a "sentiment classifier"(Sec. 2.1.2) is employed to extract the user *sentiment* (⑤). In this case, the extracted sentiment is *"Neutral"*

---

[1]Note: there exist other large language models, such as GPT [7] from OpenAI and LaMDA [8] from Google. However, they are even more resource-hungry than BERT (thus less suitable for embedded deployment), and some are close-sourced. Hence, our initial study limits on BERT-based architectures.

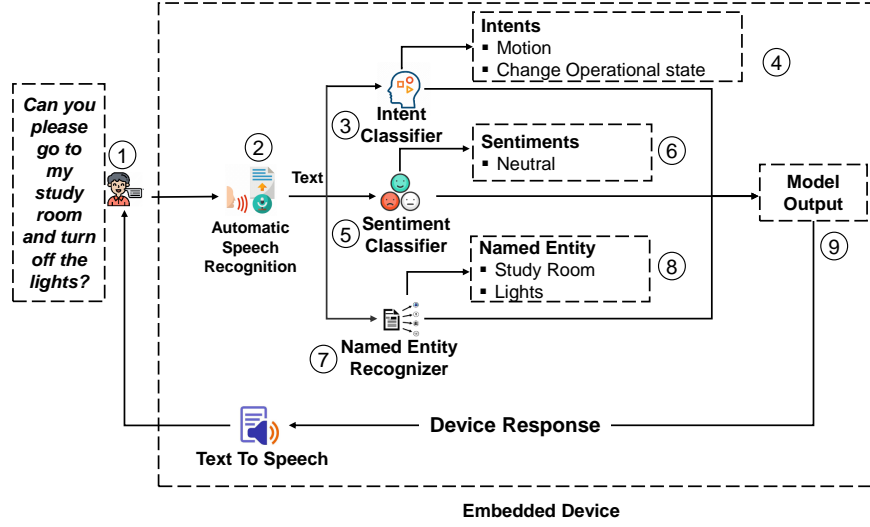[2]Section 2.1 formally presents these three NLP tasks.

Figure 1: Utterance processing steps of a voice-controlled embedded device.

as the command does not express any emotion (⑥). Sentiment classification helps the voice-controlled device grasp the emotional context behind user utterances, enabling it to respond appropriately. In addition, the dialog system utilizes a "named entity recognizer" (Sec. 2.1.3) to identify specific "entities" (⑦-⑧). NER is crucial for accurately identifying user-specified entities like locations and objects, ensuring precise execution of commands in this scenario. For example, the entities, in this case, are *"study room"* (location) and *"lights"* (object). Once the user's intention and relevant entities are identified (⑨), the control application running on the embedded device carries out the specified task and sends a response back to the user. In this paper, we focus on understanding how the language models perform on embedded platforms for IC, SC, and NER tasks (e.g., steps ③-⑧).

2.1 NLP Tasks under Consideration

Recall from our earlier discussion that intent/sentiment classification and named-entity recognition are fundamental NLP tasks for any voice-controlled interactive system, chatbots, and virtual assistants. We now formally introduce IC, SC, and NER tasks.

2.1.1 Intent Classification. To produce an accurate response, reduce backtracking, and minimize user frustration, Intent Classification (IC) is needed to identify which subsequent action a robot needs to perform depending on the user's utterance. A formal definition of IC can be given as follows:

DEFINITION 1. *Given a collection of user utterances U = $\{u_1, u_2, ..., u_n\}$, and a set of intent labels $I_x = \{i_1, i_2, ..., i_m\}$, classify each utterance $u_j \in U$ with one to more intents labels from I_x.*

Importantly, a user might have more than one intent while speaking to a robot and understanding the implicit or explicit intent expressed in a statement is essential to capturing the user's needs. For example, consider the following command: "*Can you please go to my study room and turn off the lights?*" The command wants

the robot to turn off the lights in the study room; the relevant intent here is "*Change Operational State*". However, the statement also expressed another intent related to "*Motion*", as the command requires the robot to change location. Without identifying all the underlying intents, the system cannot perform the right next step. Hence, recognizing and understanding *all* types of intents stated in an utterance is crucial for accomplishing the eventual goal.

2.1.2 Sentiment Classification. Sentiment analysis is regarded as an important task for accurate user modeling in natural dialog-based interactions, where user utterances are usually classified into multiple emotion/sentiment labels. A formal definition can be given as follows:

DEFINITION 2. *Given a collection of user utterances U = $\{u_1, u_2, ..., u_n\}$, and a set of sentiment labels $S_x = \{s_1, s_2, ..., s_m\}$, classify each expression $u_j \in U$ with one to more sentiment labels from S_x.*

For example, the following user utterance, *"OMG, yep!!! That is the final answer. Thank you so much!"* will be classified with sentiment labels "gratitude" and, "approval". Similarly, statements such as *"This caught me off guard for real. I'm actually off my bed laughing"* will be labeled as "surprise" and "amusement."

2.1.3 Named-entity Recognition. Named entity recognition (NER) — often referred to as entity chunking, extraction, or identification — is a sub-task of information extraction that seeks to locate and classify named entities mentioned in unstructured text. An entity can be expressed by a single word or a series of words that consistently refer to the same thing. Each detected entity is further classified into a predetermined category. The formal definition of the NER task can be given as follows:

DEFINITION 3. *Given a collection of statements/texts S = $\{s_1, s_2, ..., s_n\}$, and a set of entity labels $E_x = \{e_1, e_2, ..., e_m\}$, all the words/tokens in the text will be classified with an entity label $e_i \in E_x$.*

NER can be framed as a sequence labelling task that is performed in two steps, first, detecting the entities from the text, and second, classifying them into different categories. A named entity recognizer model classifies each word/phrase representing an entity into one of the four types: *(a)* persons (PER), *(b)* objects (OBJ), *(c)* locations (LOC), and *(d)* miscellaneous names (MISC).

2.2 Datasets

Our study includes the following datasets: *(a)* HuRIC (for IC), *(b)* GoEmotion (for SC), and *(c)* CoNLL and WNUT17 (for NER), as we present below.

2.2.1 Intent Classification: HuRIC. For IC, we use Human Robot Interaction Corpus (HuRIC) [16], which is the state-of-the-art single-class classification dataset. The basic idea of HuRIC is to build a reusable corpus for human-robot interaction in natural language for a specific application domain, i.e., house service robots. HuRIC includes a wide range of user utterances given to a robot representing different situations in a house environment. The motivation behind selecting HuRIC for our intent classification task stems from our specific interest in utilizing a dataset that captures human-robot conversations. The HuRIC dataset allows us to train and evaluate intent classification models on realistic dialogues between humans and robots in real-world scenarios. Table 1 presents some statistics of HuRIC.

Statistic	Count
Number of examples	729
Number of intent labels	11
Size of training dataset	583
Size of test dataset	146

Table 1: Statistics of HuRIC dataset.

2.2.2 Sentiment Classification: GoEmotion. We use GoEmotion [6] dataset from Google AI for the SC task. GoEmotion is a human-annotated dataset of 58,000 Reddit comments extracted from popular English-language subreddits and labeled with 27 emotion categories. As the largest fully annotated English language fine-grained emotion dataset to date, the GoEmotion taxonomy was designed with both psychology and data applicability in mind Table 2 presents some statistics of GoEmotion. We chose the GoEmotion dataset for SC task to ensure rigorous testing of our models. With its comprehensive emotion coverage and nuanced labeling, GoEmotion serves as a challenging yet realistic benchmark for evaluating sentiment detection performance.

2.2.3 Named-entity Recognition: CoNLL & WNUT17. For NER we consider two datasets, viz., CoNLL [17] and WNUT17 [5].

CoNLL. CoNLL-2003 [17] was released as a part of CoNLL-2003 shared task: language-independent named entity recognition. The English corpus from this shared task consists of Reuters news stories between August 1996 and August 1997, each annotated with the entities associated with them. The data set consists of a training file, a development file, and a test file. The details of CoNLL-2003 are presented in Table 3.

Statistic	Count
Number of labels	27 + Neutral
Maximum sequence length in overall datasets	30
Size of training dataset	43,410
Size of test dataset	5,427
Size of validation dataset	5,426

Table 2: Statistics of GoEmotion dataset.

Statistic	Articles	Sentences	Tokens
Training set	946	14,987	203,621
Development set	216	3,466	51,362
Test set	231	3,684	46,435

Table 3: Statistics of CoNLL dataset.

WNUT17. While the CoNLL corpus is based on news stories, we wanted to select a dataset that contains user utterances such as those available on HuRIC. Unfortunately, we could not find such a NER dataset but discovered a very similar corpus (WNUT2017 [5]) that contains user-generated text. The WNUT2017 dataset's shared task focuses on identifying unusual, previously-unseen entities in the context of emerging discussions. Identifying entities in noisy text is really challenging, even for human annotators, due to novel entities and surface forms. In this dataset, user comments were mined from different social media platforms because they are large, and samples can be mined along different dimensions, such as texts from/about geo-specific areas, about home aid, and particular topics and events. Table 4 summarizes the dataset properties.

Statistic	Count
Number of examples	5690
Number of labels	6
Size of training dataset	3394
Size of test dataset	1287
Size of validation dataset	1009

Table 4: Statistics of WNUT17 dataset.

3 EXPERIMENTAL SETUP

We now summarize BERT architectures and configurations used in our experiments (Sec. 3.1 and Sec. 3.2). We selected four off-the-shelf platforms from different chip vendors (Intel, AMD, ARM, NVIDIA) to understand the feasibility of using NLP tasks on a variety of architectures (Sec. 3.3). To measure the performance of each task (IC, SC, NER), we use three popular metrics (i.e., Precision, Recall, and F_1 score). Section 3.4 lists the design questions explored in our investigation. The blueprints of our implementation, including related code/documentation, is **publicly available** for community use [13].

	BERT	RoBERTa	DistilBERT	TinyBERT
Number of Layers	12	12	6	4
Attention Heads	12	12	12	12
Hidden Layer Size	768	768	768	312
Feed-Forward Layer Size	3072	3072	3072	1200
Vocabulary Size	30522	30522	30522	30522

Table 5: Attributes of the BERT variants used in our study.

3.1 Off-the-Shelf BERT Variants

We use a pre-trained base variant of BERT [18], RoBERTa [19], DistilBERT [3], TinyBERT [4] model from Huggingface[3] and finetune the models on respective datasets (i.e., HuRIC, GoEmotion, CoNLL, and WNUT17). Table 5 listed the parameters of the BERT variants and Table 6 presents the hyper-parameter used in our experiments.

Hyperparameter	NER	IC/SC
Number of epochs	3	3
Batch size	64	64
Learning rate	$[e^{-6}, e^{-4}]$	$[e^{-6}, e^{-4}]$
Weight decay	$[0.01, 0.3]$	$[0.01, 0.3]$
Optimizer	Adam	Adam
Adam epsilon	$1e^{-8}$	$1e^{-8}$
Max sequence length	64	128

Table 6: Hyperparameter values for finetuning BERT on IC/SC and NER tasks.

3.2 Pruning and Custom Configurations

We also experiment with custom, smaller BERT configurations. Due to the resource constraints (e.g., memory and energy limits) of embedded devices, it is necessary to explore different variants of BERT-based models that can be optimized to run on the device. We can reduce the model size on two fronts: *(a)* by reducing the layer size and *(b)* by pruning various attributes.

In our study, we experiment with *different layer combinations* of BERT models and test their performance on *different hardware configurations*, which are presented in Table 12, and 13. With two layers of $BERT_{Base}$ (instead of 12), the model size reduces significantly, but so does the accuracy (in terms of F_1 score). Still, these models give better accuracy than the distilled models on complex NLP tasks. Also, where $BERT_{Base}$ model with 12 layers cannot run on a resource-constrained device, using a lesser number of layers enable a model to execute on tiny devices with good accuracy compared to the distilled methods (DistilBERT and TinyBERT).

For further shrinking of the model, pruning can be applied to weights, neurons, layers, channels, and attention heads, depending on the heuristic used. In this paper, we focus on pruning *attention heads*, which plays an important role in the model's performance

---

[3]https://huggingface.co/.

and contributes a large number of parameters. Although multi-headed attention is a driving force behind many recent state-of-the-art models, Michel et al. [20] finds that even if models have been trained using multiple heads, in practice, a large percentage of attention heads can be removed without significantly impacting performance. In fact, in some layers, *attention heads* can even be reduced to a single head.

Based on this fact, we experiment with reducing the size of $BERT_{Base}$ by dynamically pruning the *attention heads*. Each attention head provides a distribution of attention for the given input sequence. For each attention head, we calculate the head importance by calculating the *entropy* of each head. After that, we mask all the attention heads with the lowest entropy. This process is repeated for each layer of the encoder. After masking the heads, we calculate the overall F_1 score of the masked model and determine the drop in F_1 score compared to the original unpruned model. If the drop is less than a predefined threshold, we prune the masked attention heads. We repeat the process until the drop in F_1 score reaches the predefined threshold. This pruning procedure reduces the model size significantly while maintaining the desired model performance.

3.3 Evaluation Platforms

We evaluate the BERT models on heterogeneous setup, viz., x86 (Intel and AMD) and ARM platforms, including devices with or without GPU. In particular, we used the following four different embedded platforms: *(a)* Raspberry Pi 4 Model B [9], *(b)* Jetson Nano [10], *(c)* UP[2] [11], and *(d)* UDOO Bolt [12]. Table 7 lists the hardware configurations used in our setup. Among them, only Jetson board equipped with GPU (128 NVIDIA CUDA cores). We used 2 GB and 4 GB memory configurations for each of the four boards. The SoC (System on Chip) of ARM-based boards (Raspberry Pi and Jetson) configured with soldered memory — hence, we used two different boards for each with 2 GB and 4 GB RAM configurations. The x86-based boards (i.e., UP[2] and UDOO) are modular, so we used the same board but two different DDR4 RAMs (2 GB and 4 GB). Hence, our evaluation setup consists of *eight different hardware configurations* running on four ARM and two x86 boards. For energy measurements during the inference steps, we used UM25C energy meter [21]. We performed all experiments on Linux kernel 5.15.0. The NLP models were developed using PyTorch library (version 1.13).

3.4 Design Challenges & Research Questions

We conducted extensive experiments to investigate the following research questions (RQs).

- **RQ 1.** Given specific user-defined constraints, such as system resources (processor, memory) and performance budgets (accuracy, inference time), what is the optimal (if any) BERT-based architecture satisfying those constraints?
- **RQ 2.** What is the accuracy vs. model-size trade-off as we prune the models?
- **RQ 3.** What are the trade-offs of accuracy and corresponding resource usage (e.g., memory, inference-time, energy consumptions) as we perform pruning?
- **RQ 4.** Does GPU aid in inference time?

Embedded Platform	Architecture	CPU	GPU	Memory
Raspberry Pi	ARM	Quad-core Cortex-A72	✖	2 GB and 4 GB
Jetson Nano	ARM	Quad-core Cortex-A57	128-core NVIDIA Maxwell	2 GB and 4 GB
UP^2	x86	Dual-core Intel Celeron N6210	✖	2 GB and 4 GB
UDOO Bolt	x86	Quad-core AMD Ryzen V1605B	✖	2 GB and 4 GB

Table 7: Embedded platforms used in our evaluations: *(a)* **Raspberry Pi [9],** *(b)* **Jetson Nano [10],** *(c)* **UP^2 [11], and** *(d)* **UDOO Bolt [12]. We used 2 GB and 4 GB memory configurations for each board.**

- **RQ 5.** What are the energy consumption differences among various architectures (x86 and ARM)? Does the presence of GPUs impact energy usage?

4 RESULTS

In this section, we report our results for the three basic NLP tasks, i.e., IC, SC, and NER for both existing (e.g., BERT, RoBERTa, DistilBERT, and TinyBERT) and custom BERT architectures.

4.1 Experience with Existing BERT Variants

Intent Classification (IC). Recall from our earlier discussion (Sec. 2.2.1) that we used the HuRIC dataset for IC. Table 8 presents our findings after running IC tasks using the HuRIC dataset on different hardware. We observed that all the models performed similarly on this dataset, achieving more than 90% F_1 Score.

Multi-label Sentiment Classification (SC). We next analyze the performance of multi-label SC tasks on Raspberry Pi, Jetson, UP^2, and UDOO board. As mentioned in Sec. 2.2.2, we used GoEmotion [6] dataset for this task. GoEmotion includes direct user-to-user conversation text and labels them with many user emotions. Table 9 summarizes the performance of all the models for this task. Interestingly, for this task, DistilBERT and TinyBERT failed drastically, as they achieved a very low F_1 Score. The failure of distilled models can be attributed to the difficulty of the task. Multi-label SC requires each utterance to be classified with more than one sentiment. Therefore, this dataset is not a straightforward positive-negative sentiment detection.

Named-entity Recognition (NER). As we mention in Sec. 2.2.3, we use two different datasets to test the NER task. Table 10 summarizes the performance over both the NER datasets and shows that for the CoNLL dataset. In this setup, all models performed comparatively the same. However, the performance of distilled models dropped sharply for the WNUT17 dataset (which focuses on identifying unusual, previously-unseen entities). This drop tends to be due to

the difficulty of analyzing this task, as NER evaluates the ability to detect and classify novel, emerging, singleton-named entities in noisy inputs.

In summary, our findings are as follows.

- All models achieved decent F1 scores (>90%) for IC task.
- DistilBERT and TinyBERT struggled with the multi-label SC task as none achieved an F1 score of more than 15%.
- All BERT models excelled in the NER task on the CoNLL dataset and accurately recognized named entities (resulting in >90% F1 scores).
- Distilled models showed a performance decline for the NER task on the WNUT17 dataset with the F1 score dropping to less than 5%, indicating difficulty with dataset intricacies.

4.2 Exploration with Custom Architectures

Based on our experiment results (Tables 8–10), we further explore different alternative BERT-based architectures by reducing the layers and pruning the attention heads from the original $BERT_{Base}$ model. For this exploration, we primarily focus on the challenging tasks, i.e., multi-label SC and NER where off-the-shelf models (e.g., DistilBERT and TinyBERT) failed to perform.

Table 11–Table 13 present our exploration findings.[4] We discuss our observations in Sec. 4.2.3 and provide answers to the research questions posed in Sec. 3.4. Before we proceed with the discussion, we present a brief overview of the attributes and metrics used in our evaluation.

4.2.1 Model Attributes. In the evaluation, we vary the following model attributes.

- F_1 *Threshold* (θ): The θ-cells represents what percentage of the F_1 score (with respect to $BERT_{Base}$) is retained by the models. In our experiment, we varied θ between 50% to 90% and reported

---

[4]**Note:** We omit the results for IC on custom BERT architectures as existing models suffice to perform this task.

Intent Classification Task (Dataset: HuRIC)

BERT			RoBERTa			DistilBERT			TinyBERT		
Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
0.943	0.985	0.961	0.975	0.952	0.962	0.951	0.903	0.927	0.912	0.897	0.902

Table 8: Performance of BERT, RoBERTa, DistilBERT, and TinyBERT for the IC task.

Multi-label Sentiment Detection Task (Dataset: GoEmotion)

BERT			RoBERTa			DistilBERT			TinyBERT		
Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
0.77	0.37	0.490	0.731	0.452	0.567	0.174	0.121	0.134	0.060	0.030	0.031

Table 9: Performance of BERT, RoBERTa, DistilBERT, and TinyBERT for the SC task.

Named-entity Recognition Task

Dataset: CoNLL

BERT			RoBERTa			DistilBERT			TinyBERT		
Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
0.891	0.963	0.926	0.882	0.955	0.917	0.906	0.967	0.935	0.872	0.958	0.911

Dataset: WNUT17

BERT			RoBERTa			DistilBERT			TinyBERT		
Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
0.671	0.295	0.410	0.537	0.315	0.397	0.316	0.014	0.028	0.0	0.0	0.0

Table 10: Performance of BERT, RoBERTa, DistilBERT, and TinyBERT for the NER task.

| Task | Metrics | F_1 Score Threshold (θ) | | | | | | | | | | | | | | | | | | |
		θ_{50}				θ_{60}				θ_{70}				θ_{80}				θ_{90}			
		2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
IC	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	MS	144.3	N/A	N/A	N/A	148.3	N/A	N/A	N/A	N/A	195.6	N/A	282.2	154.2	198.7	246	N/A	N/A	211.3	268	303.5
	Params	35.1				37.1					48.9		70.5	38.6	49.7	61.5			52.8	67	75.9
	Pruning	68%				66%					56%		36%	65%	55%	44%			52%	39%	31%
NER	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	MS	136.4	N/A	N/A	N/A	147.4	N/A	N/A	N/A	N/A	185.2	233.3	N/A	N/A	201	249.8	268	N/A	N/A	260.8	289.2
	Params	34.1				36.9					46.3	58.3			46.3	62.4	67			65.2	72.3
	Pruning	67%				65%					57%	45%			45%	57%	38%			39%	32%

Table 11: Performance of SC and NER tasks for the GoEmotion and WNUT17 datasets on various configurations. In metrics column, MS= Model Size (MB), Params= Parameters (Million).

the model details in respective columns. For example, θ set to 80% implies the θ_{80} column.

- *Platform*: Indicates the memory capacity of the different hardware we used in our exploration.
- *Layer*: Represents the number of layers retained.
- *Model Size*: The size of the saved model after training. We stored the saved model on the disk which is then loaded on the memory for inference.
- *Parameters*: This metric indicates the total number of parameters in the saved model. We obtained the model parameters using the `model.parameters()` method in PyTorch [22].
- *Pruning*: Pruning percentage represents the reduction in size from the $BERT_{Base}$ model. For example, a pruning percentage of 70% implies that the pruned model is 70% smaller than $BERT_{Base}$.

4.2.2 Performance Metrics. We consider the three metrics to benchmark the NLP models: *(a)* inference time, *(b)* memory usage, and *(c)* energy consumption, as we present below.

- *Memory Consumption*: Maximum memory usage (in megabytes) of the corresponding NLP task running on different boards during the inference time. We used Python `memory_profiler` for each input to get the memory usage.
- *Inference Time*: Depending on the specific task, the 95th percentile time required for the model to infer the appropriate *Intent, Sentiment* or *Entity* from a given command.
- *Energy Consumption*: The average energy consumed (in watts) by the different hardware during the inference of a given command. We measured both the *rest time* (i.e., when the system is idle) and *inference time* energy consumption.

Plat.	Metrics	θ_{50}				θ_{60}				θ_{70}				θ_{80}				θ_{90}			
		2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
Pi 2 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	EC	4.24	N/A	N/A	N/A	4.28	N/A	N/A	N/A	N/A	4.32	N/A	4.37	4.35	4.3	4.48	N/A	N/A	3.91	4.45	4.34
	MC	394.3				391.5					440.4		563	401.9	446.2	497.3			462.9	521.5	559.4
	IT	0.52				0.51					1.1		2.17	0.54	1.01	1.69			1.3	1.81	2.35
Pi 4 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	N/A	4	6	8
	EC	4.91	N/A	N/A	N/A	5.05	N/A	N/A	N/A	N/A	4.67	N/A	5.09	4.98	4.72	4.8	N/A		4.77	4.78	4.95
	MC	397.5				401					446.3		569.2	404.2	453.2	507.7			464.6	528.8	563.4
	IT	0.44				0.44					0.82		1.5	0.5	0.81	1.21			1.00	1.46	1.55
Jetson 2 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	EC	5.87	N/A	N/A	N/A	5.96	N/A	N/A	N/A	N/A	6.01	N/A	6.28	5.87	6.14	6.26	N/A	N/A	6.13	6.22	6.24
	MC	299.2				320.1					345.4		478.5	329.7	352.6	415			367.5	429.8	463.9
	IT	0.29				0.26					0.49		0.88	0.30	0.51	0.68			0.56	.797	1.02
Jetson 4 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	EC	6.27	N/A	N/A	N/A	6.25	N/A	N/A	N/A	N/A	6.21	N/A	6.43	6.27	6.31	6.36	N/A	N/A	6.39	6.4	6.47
	MC	355.5				359.7					410.1		532.9	366	408.8	459.4			423	488.4	524
	IT	0.27				0.27					0.45		0.86	0.30	0.46	0.66			0.50	0.76	0.87
UP² 2 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	EC	10.897	N/A	N/A	N/A	10.9	N/A	N/A	N/A	N/A	10.94	N/A	10.79	10.96	11.05	10.78	N/A	N/A	10.81	10.73	10.76
	MC	670.9				708.6					606.8		741.7	696.8	693	742.6			741.1	707.4	782.1
	IT	0.12				0.12					0.21		0.39	0.13	0.21	0.30			0.23	0.31	0.42
UP² 4 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	EC	10.17	N/A	N/A	N/A	11.05	N/A	N/A	N/A	N/A	10.04	N/A	11.51	10.99	11.20	11.38	N/A	N/A	11.01	11.38	11.16
	MC	738.3				653.2					715.7		870	736.8	753.4	792.9			736.2	798.3	875.3
	IT	0.12				0.12					0.21		0.40	0.13	0.22	0.30			0.24	0.30	0.41
UDOO 2 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	EC	23.08	N/A	N/A	N/A	23.12	N/A	N/A	N/A	N/A	23.43	N/A	23.64	23.28	23.38	23.62	N/A	N/A	23.57	23.56	23.46
	MC	379.3				424.6					361.5		422	377.9	348.1	461			360.9	468.2	453.1
	IT	0.06				0.06					0.10		0.12	0.06	0.09	0.13			0.10	0.14	0.17
UDOO 4 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	EC	23.076	N/A	N/A	N/A	23.12	N/A	N/A	N/A	N/A	22.78	N/A	23.18	22.98	23.05	22.99	N/A	N/A	23.13	22.96	23.15
	MC	450.1				445.4					487.3		596.8	443.8	505.9	547.8			505.7	544.7	601.9
	IT	0.07				0.07					0.11		0.16	0.08	0.11	0.15			0.12	0.16	0.16

Table 12: Performance of SC task for the GoEmotion dataset on various configurations. In metrics column, EC=Energy Consumption (W), MC= Memory (MB), and IT= Inference Time (s). The gray cells highlight the best-case scenario for each F_1 threshold (θ).

4.2.3 Observations.
We now discuss our major observations and address the research questions introduced in Sec. 3.4.

Selecting "suitable" model subject to given constraints [RQ 1]. We can address this specific research question by inspecting Table 12 and 13. Note that Table 12 and 13 provide information on the model size, performance, parameters, and pruning for the SC and NER tasks, respectively. Let us assume a system designer is looking for suitable NER models for a 2 GB embedded platform that maintains approximately 70% of BERT's accuracy (θ_{70}). In this case, we can *(a)* scan through the NER performance metrics (i.e., Table 13), and *(b)* observe from Pi 2 GB *Platform* row and θ_{70} column that a six-layered and pruned (45% reduced) BERT model can run on a Pi 2 GB platform and attain 70% of $BERT_{Base}$'s original F_1 score. Hence, our exploration (and similar experiments along this line) can *aid the designers to select appropriate models with desired performance guarantees.*

Accuracy and model-size trade-offs for pruned architectures [RQ 2]. Table 12 and Table 13 further provide insights on the pruning vs. F_1 score trade-off. For example, in Table 12, the Pi 2 GB *Platform* row shows a set of models that can run on that system. The same row and θ_{80} (80% F_1 *score threshold*) column show that even pruning 55% of a $BERT_{Base}$ model with four layers can retain 80% of

original F_1 score of $BERT_{Base}$, while the model size can be reduced to 198.7 MB from 441.55 MB. Tables 12 and 13 indicate that although *pruning has only a minor impact on memory consumption, it does not have a significant effect on energy consumption.* Furthermore, our analysis of Table 12 suggests that *inference time is directly proportional to the size of the model, implying that decreasing the model size leads to a decrease in inference time.*

Accuracy vs. system resource trade-offs for pruned architectures [RQ 3]. If a user has precise requirements for inference time and memory consumption for a given hardware, one can scan Table 12 and 13 to pick the optimum model that meets those requirements. For instance, if we want to find NER models that can make inferences in less than 0.56 seconds on Raspberry Pi that has 4 GB of memory, the corresponding *Platform* row Pi (4 GB) in Table 13, shows us the model parameters that can satisfy this requirement(e.g., two-layered, four-layered). Since both of them are feasible for the chosen platform, designers can choose any of them based on the required application performance. As an example, if we pick a two-layered BERT model, the accuracy is 60%, and the memory consumption is 698.3 MB. In contrast, if we select the four-layered BERT model, it can achieve 80% accuracy, with a cost of higher memory consumption of 699.1 MB. Hence, at the expense of slightly higher memory consumption, it is possible to get 20%

Plat.	Metrics	θ_{50}				θ_{60}				θ_{70}				θ_{80}				θ_{90}			
Pi 2 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	EC	4.28	N/A	N/A	N/A	4.36	N/A	N/A	N/A	N/A	4.3	4.51	N/A	N/A	3.95	4.35	4.51	N/A	N/A	4.38	4.53
	MC	676.3				709					721.3	678.4			736.8	700.3	678.4			704.7	692.2
	IT	0.55				0.55					1.08	0.61			1.19	0.61	0.61			0.56	0.65
Pi 4 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	EC	4.48	N/A	N/A	N/A	4.53	N/A	N/A	N/A	N/A	4.62	4.63	N/A	N/A	4.59	4.85	4.82	N/A	N/A	4.9	4.88
	MC	675				698.3					683.6	698.9			699.1	706	686			706.6	709
	IT	0.49				0.516					0.57	0.57			0.55	0.57	0.63			0.60	0.65
Jetson 2 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	EC	5.74	N/A	N/A	N/A	5.8	N/A	N/A	N/A	N/A	5.96	5.7	N/A	N/A	6.05	5.77	5.7	N/A	N/A	5.8	5.71
	MC	314.6				348.2					388.7	357.7			392.4	359	362.7			362.3	368.9
	IT	0.29				0.29					0.45	0.29			0.50	0.31	0.29			0.33	0.29
Jetson 4 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	EC	5.74	N/A	N/A	N/A	5.76	N/A	N/A	N/A	N/A	6.23	5.95	N/A	N/A	6.04	6.12	6.04	N/A	N/A	6.05	6.08
	MC	368.3				365.8					418.5	368.7			424.4	365.5	361.7			363.5	366.6
	IT	0.29				0.27					0.497	0.29			0.45	0.33	0.29			0.32	0.30
UP^2 2 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	EC	11.108	N/A	N/A	N/A	11.2	N/A	N/A	N/A	N/A	11.18	11.44	N/A	N/A	11.21	10.77	10.72	N/A	N/A	10.63	10.7
	MC	512.5				597.6					652.9	574			650.8	601.8	603.5			591.3	591.1
	IT	0.13				0.12					0.19	0.18			0.20	0.12	0.12			0.12	0.13
UP^2 4 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	EC	10.99	N/A	N/A	N/A	11.21	N/A	N/A	N/A	N/A	11.36	11.33	N/A	N/A	11.32	11.40	11.35	N/A	N/A	11.16	11.3
	MC	599.8				601.8					651.4	658.1			653.7	652.7	657.7			657.7	652.9
	IT	0.12				0.11					0.19	0.20			0.19	0.19	0.22			0.22	0.21
UDOO 2 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	EC	24.02	N/A	N/A	N/A	24.08	N/A	N/A	N/A	N/A	24.09	24.07	N/A	N/A	24.14	24.06	24.02	N/A	N/A	24.11	24.07
	MC	437.8				438.6					491.2	490.8			490.9	490.5	503.4			490.3	492.4
	IT	0.07				0.05					0.09	0.09			0.09	0.09	0.09			0.09	0.08
UDOO 4 GB	Layer	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8	2	4	6	8
	EC	22.79				22.86					23.07	23.57			23.09	23.49	23.39			23.58	23.50
	MC	445.6				443					500.1	496.4			497.5	499.5	510.1			495.2	499.8
	IT	0.06				0.05					0.08	0.09			0.09	0.08	0.09			0.08	0.08

Table 13: Performance of NER task for the WNUT17 dataset on various configurations. In metrics column, MS= Model Size (MB), Params= Parameters (Million), EC=Energy Consumption (Watt), MC= Memory (MB), and IT= Inference Time (s). The gray cells highlight the best-case scenario for each F_1 threshold (θ).

more accuracy. Such a lookup-based approach allows the designers to perform a desired *cost-benefit analysis*.

The case for GPUs [RQ 4]. Intuitively, GPUs aid in any learning-enabled tasks. We used one GPU-enabled hardware (Jetson Nano) in our design space exploration. As Table 12 and Table 13 illustrate Jetson reduces inference time compared to the Raspberry Pi board (no GPU). However, GPUs alone cannot provide faster inference. For instance, x86 boards (UP^2 and UDOO) do not have GPUs but are equipped with a faster processor, and hence result in better inference times (i.e., took less time to process the queries). Systems with better hardware (CPU/GPU/memory) can output inference decisions faster, which may increase power consumption, as we discuss next.

Energy consumption on various architectures and model configurations [RQ 5]. Since many embedded platforms used for NLP tasks (e.g., voice-controlled robots, voice assistants, IoT devices) are battery-operated, energy consumption for inferring user commands is a crucial parameter. Hence, we also analyze the energy usage of the NLP tasks. For any selected BERT model, one can find the system energy consumption from Table 12 and 13, for two different tasks, respectively. As the table shows, (for a given hardware) during the inference of a given command, *energy consumption does not vary significantly for various models.*
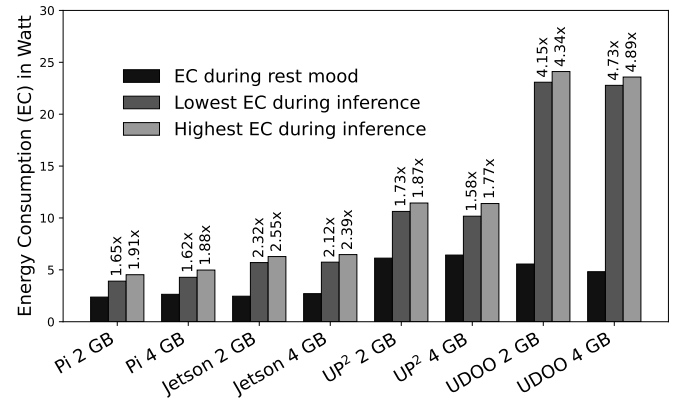
Figure 2: Energy consumption during rest mode and inference period. ARM devices use less energy compared to x86 systems.

To understand the energy usage of the various NLP tasks on our test platforms, we also measured the energy consumption of each board during rest mode and inference period (see Fig. 2). We obtained the rest mode energy usage by idling the device

for 10 minutes and took the average value. For inference energy consumption, we tested with 40 SC and 40 NER queries and repeated each of them 100 times (i.e., a total of $2 \times 40 \times 100 = 8000$ samples. We report the maximum (lighter gray) bar and minimum (darker gray) energy consumption values of the 8000 trials.

As Fig. 2 shows, the inference energy consumption increases by a factor of 1.73 to 4.89 times compared to the rest mode energy usage. Besides, ARM architectures (Raspberry Pi and Jetson) consume less energy than the x86 architectures (UP^2 and UDOO). Another interesting observation is that even though Jetson boards use GPU, they are more power efficient for performing NLP tasks compared to some CPU-only x86 systems. Our experiments show that the AMD Ryzen platform (UDOO) performs poorly in terms of energy usage. However, as Table 12 and Table 13 indicate, UDOO boards output faster inference time (since they have relatively faster CPU than the others). Hence, there exists a *trade-off* between inference time and energy usage.

4.2.4 Summary of Findings. Our key findings for custom BERT architectures are listed below.

Model Size & Pruning.
- Pruning helps in reduction in size (upto 67%) while maintaining at least 50% of $BERT_{Base}$'s F1 score.
- The time required for inference is directly related to the size of the model, i.e., a smaller model size results in a reduction in inference time.
- Pruning of attention heads does not reduce memory usage.
- Pruning attention heads does not improve energy consumption significantly.

System Artifacts.
- Faster x86 (Intel and AMD) platforms outperform ARM SoCs (e.g., Pi and Jetson boards) wrt. inference time, but their energy consumption is significantly higher (e.g., at least 2.60 times) than ARM counterparts.
- GPUs aid in performance (e.g., inference times are approx. 2-times faster in Jetson than Raspberry Pi) but GPUs alone in Jetson boards cannot outperform a relatively faster CPU (i.e., those used in UP^2 and UDOO boards).
- Powerful processor can decrease inference time (as expected) but comes with a cost (increased power consumption: 2.60-5.90 times higher).

---

5 DISCUSSION

We explore different custom architectures of BERT-based language models and test their deployment feasibility in low-power embedded devices. We conducted extensive performance evaluations on four embedded platforms from various vendors with varying computing capabilities to cover a wide range of application scenarios. We show that it is *not always feasible to shrink* the size of "finetuned" $BERT_{Base}$ model that can satisfy specific user-defined accuracy/performance budgets. We also report which models are deployable to resource-constrained devices with given user requirements. We believe our empirical findings will help the developers quickly narrow down the plausible BERT-based architecture for target applications, thus saving development

time and effort. While we tested the NLP models on four embedded platforms (a total of 8 hardware configurations) in a Linux environment, they can be ported to other systems, such as smartphones/tablets running different OSes (such as Android). Thus our empirical study is applicable in broader human-centric application domains, including chatbots [23–25], virtual assistants [26, 27], and language translation [28, 29].

Our study is limited to BERT-based models for four existing datasets (i.e., may not generalize to other language models and datasets). However, our evaluation framework is modular and can be retrofitted to other architectures/datasets without loss of generality. While shrinking models have made it possible to deploy them on resource-constrained embedded devices, their performance on new datasets or tasks is often limited. One potential solution to mitigate this issue is to utilize continual learning techniques [30, 31], as they allow models to continuously learn and evolve based on increasing data input while retaining previously acquired knowledge. Our future work will explore the feasibility of employing continual learning for embedded devices.

One of the challenges to figuring out the *optimal* BERT-based architecture is the lack of application-specific (viz., voice-controlled robots for home automation) datasets. Existing datasets either *(a)* do not have enough examples for training deep learning models or *(b)* do not provide complex, practical queries to test the robustness of a given model. Building suitable datasets for IoT-specific human-centric applications such as voice-control home/industrial automation is an interesting open research problem.

6 RELATED WORK

We discuss related research on two fronts: *(a)* BERT-based models and their efficient variants and *(b)* using NLP on embedded devices.

6.1 BERT-based Models and their Variants

The performance of BERT comes at a high computation and memory cost, which makes on-device inference really challenging. To mitigate this issue, researchers have proposed knowledge distillation approaches from the original BERT model, for example, *(a)* "finetune" the BERT model to improve task-specific knowledge distillation [32, 33], *(b)* use Bi-LSTM models [34] for knowledge distillation from BERT, *(c)* leverage single-task models to teach a multi-task model [35], *(d)* distillation of knowledge from an ensemble of BERT into a single BERT [36], *(e)* TinyBERT [4] uses a layer-wise distillation strategy for BERT in both the pre-training and fine-tuning stages, and *(f)* DistilBERT [3] halves the depth of the BERT model through knowledge distillation in the pre-training stage and an optional fine-tuning stage. On a different direction, the *Patient Knowledge Distillation* approach [37] compresses an original large model ("teacher") into an equally-effective lightweight shallow network ("student"). Other BERT models (e.g., SqueezeBERT [38], MobileBERT [39], Q8BERT [40], ALBERT [41]) can also reduce resource consumption than the vanilla BERT. EdgeBERT [42], an algorithm-hardware co-design approach, performs latency-aware energy optimizations for multi-task NLP problems. However, unlike ours, EdgeBERT *(a)* does not apply attention heads pruning, and *(b)* does not report scores on downstream NLP tasks on real-world embedded systems.

6.2 NLP for Embedded Platforms

Researchers have explored NLP techniques to facilitate natural communication between humans and embedded devices, especially in the context of voice-controlled cognitive robots. For example, Megalingam et al. [43] presents a voice recognition tool that compares the user's input commands with the stored data. Zhang et al. [44] propose a ROS-based robot that analyzes commands using an offline grammar recognition library. Megalingam et al. [45] propose a cost-efficient speech processing module running on ROS that can provide natural language responses to the user. There also exists ROS-integrated independent speech recognition packages [46, 47] as well as Arduino-based [48] and custom [49] voice-control robot platforms. House et al. [50] a voice-controlled robotic arm (named VoiceBot) for individuals with motor impairments [51]. However, most of these works focused on rule-based approaches, and we note that transformer architectures are still under-explored in terms of their practical deployment challenges in real-world embedded and robotic devices, which is the focus of this study.

6.3 Uniqueness of Our Work

While existing work can reduce the size of BERT models through distillation and pruning, from a system design perspective, it is still difficult and tedious for a developer to find out the "right" BERT-based architecture to use in an embedded platform. To date, it is also unclear which lighter version of BERT would find the optimal balance between the resources available in an embedded device (e.g., CPU, GPU, memory) and the minimum accuracy desired. We used four off-the-shelf platforms widely used by developers for various IoT and embedded applications and benchmarked state-of-the-art BERT architectures. Our empirical evaluation and design space exploration on heterogeneous platforms (e.g., x86 and ARM, with or without GPU) can help the system and machine learning engineers to pick suitable architectures depending on target system configuration and performance constraints (e.g., accuracy, F_1 score). To the best of our knowledge, this work is *one of the first efforts to study the feasibility of deploying BERT-based models* in real-world resource-constrained embedded platforms.

7 CONCLUSION

This paper presents an empirical study of BERT-based neural architectures in terms of the feasibility of deploying them on resource-constrained systems, which have become ubiquitous nowadays. Our performance evaluation results will assist developers of multiple ubiquitous computing domains, such as voice-controlled home and industrial automation, precision agriculture, and medical robots to determine the deployability of NLP models in their target platform. By using our benchmark data, designers of ubiquitous systems will now be able to select the "right" hardware, architecture, and parameters depending on the resource constraints and performance requirements. This will also save time on the developer's end, as they can make informed choices regarding which BERT-based architecture to use during development based on their NLP application scenario and the available hardware.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[2] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4171–4186. [Online]. Available: https://doi.org/10.18653/v1/n19-1423

[3] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, vol. abs/1910.01108, 2019. [Online]. Available: http://arxiv.org/abs/1910.01108

[4] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "Tinybert: Distilling BERT for natural language understanding," in *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, ser. Findings of ACL, T. Cohn, Y. He, and Y. Liu, Eds., vol. EMNLP 2020. Association for Computational Linguistics, 2020, pp. 4163–4174. [Online]. Available: https://doi.org/10.18653/v1/2020.findings-emnlp.372

[5] L. Derczynski, E. Nichols, M. van Erp, and N. Limsopatham, "Results of the WNUT2017 shared task on novel and emerging entity recognition," in *Proceedings of the 3rd Workshop on Noisy User-generated Text, NUT@EMNLP 2017, Copenhagen, Denmark, September 7, 2017*, L. Derczynski, W. Xu, A. Ritter, and T. Baldwin, Eds. Association for Computational Linguistics, 2017, pp. 140–147. [Online]. Available: https://doi.org/10.18653/v1/w17-4418

[6] D. Demszky, D. Movshovitz-Attias, J. Ko, A. Cowen, G. Nemade, and S. Ravi, "Goemotions: A dataset of fine-grained emotions," *arXiv preprint arXiv:2005.00547*, 2020.

[7] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[8] R. Thoppilan, D. D. Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H. S. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, Y. Zhou, C. Chang, I. Krivokon, W. Rusch, M. Pickett, K. S. Meier-Hellstern, M. R. Morris, T. Doshi, R. D. Santos, T. Duke, J. Soraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E. Hoffman-John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B. Aguera-Arcas, C. Cui, M. Croak, E. H. Chi, and Q. Le, "Lamda: Language models for dialog applications," *CoRR*, vol. abs/2201.08239, 2022. [Online]. Available: https://arxiv.org/abs/2201.08239

[9] R. Pi, "https://www.raspberrypi.com/," 2023, accessed:May 15. [Online]. Available: https://www.raspberrypi.com/

[10] Jetson, 2023, accessed:Sep 15. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[11] U. board, 2023, accessed:Sep 15. [Online]. Available: https://up-board.org/

[12] U. Bolt, 2023, accessed:Sep 15. [Online]. Available: https://www.udoo.org/discover-the-udoo-bolt/

[13] "Transformers bechmarking: Implementation and artifacts," 2024. [Online]. Available: https://github.com/CPS2RL/Benchmarking-BERT-Embedded-Devices-ICPE24-Artifact

[14] B.-H. Juang and L. R. Rabiner, "Automatic speech recognition–a brief history of the technology development," *Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara*, vol. 1, p. 67, 2005.

[15] D. Yu and L. Deng, *Automatic speech recognition.* Springer, 2016, vol. 1.

[16] A. Vanzo, D. Croce, E. Bastianelli, R. Basili, and D. Nardi, "Grounded language interpretation of robotic commands through structured learning," *Artificial Intelligence*, vol. 278, 2020. [Online]. Available: https://doi.org/10.1016/j.artint.2019.103181

[17] E. F. Sang and F. De Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," *arXiv preprint cs/0306050*, 2003.

[18] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[20] P. Michel, O. Levy, and G. Neubig, "Are sixteen heads really better than one?" in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 14 014–14 024. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/2c601ad9d2ff9bc8b282670cdd54f69f-Abstract.html

[21] UM25C, 2017. [Online]. Available: https://www.makerhawk.com/collections/frontpage/products/makerhawk-um25c-usb-tester-bluetooth-usb-meter-type-c-current-meter-usb-power-meter-dc-24-000v-5-0000a-usb-cable-tester-1-44-inch-color-lcd-multimeter-voltage-tester-usb-load-qc-2-0-qc-3-0

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[23] E. Adamopoulou and L. Moussiades, "Chatbots: History, technology, and applications," *Machine Learning with Applications*, vol. 2, p. 100006, 2020.

[24] P. B. Brandtzæg and A. Følstad, "Why people use chatbots," in *Internet Science - 4th International Conference, INSCI 2017, Thessaloniki, Greece, November 22-24, 2017, Proceedings*, ser. Lecture Notes in Computer Science, I. Kompatsiaris, J. Cave, A. Satsiou, G. Carle, A. Passani, E. Kontopoulos, S. Diplaris, and D. McMillan, Eds., vol. 10673.   Springer, 2017, pp. 377–392. [Online]. Available: https://doi.org/10.1007/978-3-319-70284-1_30

[25] H.-Y. Shum, X.-d. He, and D. Li, "From eliza to xiaoice: challenges and opportunities with social chatbots," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, pp. 10–26, 2018.

[26] R. W. White, "Skill discovery in virtual assistants," *Communications of the ACM*, vol. 61, no. 11, pp. 106–113, 2018.

[27] B. Schmidt, R. Borrison, A. Cohen, M. Dix, M. Gärtler, M. Hollender, B. Klöpper, S. Maczey, and S. Siddharthan, "Industrial virtual assistants: Challenges and opportunities," in *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, 2018, pp. 794–801.

[28] A. G. Oettinger, "Automatic language translation," in *Automatic Language Translation.*   Harvard University Press, 2013.

[29] P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, R. L. Mercer, and P. Roossin, "A statistical approach to language translation," in *Coling Budapest 1988 Volume 1: International Conference on Computational Linguistics.*   John von Neumann Society for Computing Sciences, Budapest, 1988.

[30] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021.

[31] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu, "Embracing change: Continual learning in deep neural networks," *Trends in cognitive sciences*, vol. 24, no. 12, pp. 1028–1040, 2020.

[32] I. Turc, M. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: The impact of student initialization on knowledge distillation," *CoRR*, vol. abs/1908.08962, 2019. [Online]. Available: http://arxiv.org/abs/1908.08962

[33] H. Tsai, J. Riesa, M. Johnson, N. Arivazhagan, X. Li, and A. Archer, "Small and practical BERT models for sequence labeling," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds.   Association for Computational Linguistics, 2019, pp. 3630–3634. [Online]. Available: https://doi.org/10.18653/v1/D19-1374

[34] R. Tang, Y. Lu, L. Liu, L. Mou, O. Vechtomova, and J. Lin, "Distilling task-specific knowledge from BERT into simple neural networks," *CoRR*, vol. abs/1903.12136, 2019. [Online]. Available: http://arxiv.org/abs/1903.12136

[35] K. Clark, M. Luong, U. Khandelwal, C. D. Manning, and Q. V. Le, "Bam! born-again multi-task networks for natural language understanding," in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, A. Korhonen, D. R. Traum, and L. Màrquez, Eds.   Association for Computational Linguistics, 2019, pp. 5931–5937. [Online]. Available: https://doi.org/10.18653/v1/p19-1595

[36] X. Liu, Y. Wang, J. Ji, H. Cheng, X. Zhu, E. Awa, P. He, W. Chen, H. Poon, G. Cao, and J. Gao, "The microsoft toolkit of multi-task deep neural networks for natural language understanding," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2020, Online, July 5-10, 2020*, A. Celikyilmaz and T. Wen, Eds.   Association for Computational Linguistics, 2020, pp. 118–126. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-demos.16

[37] S. Sun, Y. Cheng, Z. Gan, and J. Liu, "Patient knowledge distillation for BERT model compression," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds.   Association for Computational Linguistics, 2019, pp. 4322–4331. [Online]. Available: https://doi.org/10.18653/v1/D19-1441

[38] F. N. Iandola, A. E. Shaw, R. Krishna, and K. Keutzer, "Squeezebert: What can computer vision teach NLP about efficient neural networks?" in *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing, SustaiNLP@EMNLP 2020, Online, November 20, 2020*, N. S. Moosavi, A. Fan, V. Shwartz, G. Glavas, S. R. Joty, A. Wang, and T. Wolf, Eds.   Association for Computational Linguistics, 2020, pp. 124–135. [Online]. Available: https://doi.org/10.18653/v1/2020.sustainlp.1.17

[39] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "Mobilebert: a compact task-agnostic BERT for resource-limited devices," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds.   Association for Computational Linguistics, 2020, pp. 2158–2170. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-main.195

[40] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, "Q8BERT: quantized 8bit BERT," in *Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition, EMC2@NeurIPS 2019, Vancouver, Canada, December 13, 2019.*   IEEE, 2019, pp. 36–39. [Online]. Available: https://doi.org/10.1109/EMC2-NIPS53020.2019.00016

[41] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.*   OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=H1eA7AEtvS

[42] T. Tambe, C. Hooper, L. Pentecost, T. Jia, E. Yang, M. Donato, V. Sanh, P. N. Whatmough, A. M. Rush, D. Brooks, and G. Wei, "Edgebert: Sentence-level energy optimizations for latency-aware multi-task NLP inference," in *MICRO '21: 54th Annual IEEE/ACM International Symposium on Microarchitecture, Virtual Event, Greece, October 18-22, 2021.*   ACM, 2021, pp. 830–844. [Online]. Available: https://doi.org/10.1145/3466752.3480095

[43] R. K. Megalingam, R. S. Reddy, Y. Jahnavi, and M. Motheram, "Ros based control of robot using voice recognition," in *2019 Third International Conference on Inventive Systems and Control (ICISC)*, 2019, pp. 501–507.

[44] Y. Zhang, Z. Lu, C. Wang, C. Liu, and Y. Wang, "Voice control dual arm robot based on ros system."   2018 IEEE International Conference on Intelligence and Safety for Robotics (ISR), 2018, pp. 232–237.

[45] R. K. Megalingam, A. H. Kota, and V. K. T. P., "Optimal approach to speech recognition with ros," in *2021 6th International Conference on Communication and Electronics Systems (ICCES)*, 2021, pp. 111–116.

[46] Y. Zhang and S. C. Xu, "Ros based voice-control navigation of intelligent wheelchair," in *Engineering Decisions for Industrial Development*, ser. Applied Mechanics and Materials, vol. 733.   Trans Tech Publications Ltd, 4 2015, pp. 740–744.

[47] S. Sharan, T. Q. Nguyen, P. Nauth, and R. Araujo, "Implementation and testing of voice control in a mobile robot for navigation," in *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM).*   IEEE, 2019, pp. 145–150.

[48] A. Verma, D. Kumar, H. Maurya, A. Kumar, and M. P. Dwivedi, "Voice control robot using arduino," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 2, p. 04, 2020.

[49] X. Lv, M. Zhang, and H. Li, "Robot control based on voice command," in *2008 IEEE International Conference on Automation and Logistics.*   IEEE, 2008, pp. 2490–2494.

[50] B. House, J. Malkin, and J. A. Bilmes, "The voicebot: a voice controlled robot arm," in *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Boston, MA, USA, April 4-9, 2009*, D. R. O. Jr., R. B. Arthur, K. Hinckley, M. R. Morris, S. E. Hudson, and S. Greenberg, Eds.   ACM, 2009, pp. 183–192. [Online]. Available: https://doi.org/10.1145/1518701.1518731

[51] J. A. Bilmes, X. Li, J. Malkin, K. Kilanski, R. Wright, K. Kirchhoff, A. Subramanya, S. Harada, J. Landay, P. Dowden, and H. Chizeck, "The vocal joystick: A voice-based human-computer interface for individuals with motor impairments," in *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing.*   Vancouver, British Columbia, Canada: Association for Computational Linguistics, Oct. 2005, pp. 995–1002. [Online]. Available: https://aclanthology.org/H05-1125