Abstract—The accuracy of many natural language processing (NLP) tasks has been significantly improved with the development of deep learning and Transformer-based pre-trained models like BERT. However, their large number of parameters and computations pose deployment challenges. For instance, while using BERT can improve predictions in financial sentiment analysis (FSA) tasks, it can also slow down processing, where throughput and accuracy are equally important for profitability. To address these issues, we first propose an efficient and lightweight BERT model called ELBERT, which incorporates a novel confidence-window-based (CWB) early exit mechanism. In addition, an innovative approach is proposed to accelerate text processing on the GPU platform, which addresses the challenge of effectively deploying the early exit mechanism with large input batch sizes. Using these advancements, we develop a fast and high-accuracy FSA system. Experimental results demonstrate that the proposed CWB early exit mechanism outperforms existing early exit methods on BERT in terms of accuracy, at the same computation cost. By leveraging this acceleration method, our FSA system can boost the processing throughput by nearly 40 times, enabling it to process over 1000 texts per second with sufficient accuracy. This processing rate is nearly twice as fast as FastBERT, providing modern trading systems with a more powerful text processing capability.

Index Terms—Natural language processing, deep learning, model compression, financial sentiment analysis, quantitative investment, event-driven trading, BERT.

I. INTRODUCTION

SENTIMENT analysis is a rapidly evolving subfield of natural language processing (NLP) that has gained significant traction in the financial area. Its primary objective is to extract sentiment from various forms of textual data, including news articles, analyst reports, and social media posts. Financial sentiment analysis (FSA) has been extensively employed in quantitative or event-driven investment fields, where both accuracy and processing speed are crucial.

In this article, we are mainly concerned with the FSA system used in quantitative or event-driven investment fields, where both accuracy and processing speed are crucial. Nowadays, there has been a surge of quantitative investors in the market, leading to an increase in the amount of money managed by quantitative funds. As a result, the profit margins of many conventional factors have decreased, prompting investors to turn their attention to higher-dimensional data to obtain higher excess returns. Fig. 1 illustrates a modern quantitative investment system that utilizes alternative data, with text data being the most useful among them. These data are easily accessible from the Internet and have been relied on by investors for making investment decisions. Therefore, FSA, with its ability to process and mine large amounts of financial text data, has become an essential module in this system.

Although many FSA research works have been conducted, there are still problems when applying them to industry. Simple models such as dictionaries [2], [3] and SVMs [4] often have insufficient accuracy, while large deep neural networks, such as NLP Transformers [5], provide more precise results but are computationally expensive. The extensive computations significantly increase the delay of the whole investment system.

To meet the growing demand for efficient and accurate FSA with text classification, we explore model compression techniques for BERT, the most widely used NLP Transformer in this domain. Based on this exploration, a high-throughput and high-accuracy FSA system is developed. Firstly, we introduce a fast and lightweight BERT model named ELBERT, which is based on the parameter-sharing strategy from ALBERT [6]. We further improve ELBERT’s performance by incorporating our proposed confidence-window-based (CWB) early exit mechanism, a practical acceleration method suitable for Transformer neural networks [7]. Typically, early exit works best when the inference batch size equals 1. The efficiency of existing early exit methods will be influenced if the network has a large batch size, which is deployed on widely-used GPU platforms (shown in Fig. 5). Within the same batch of inputs, there are often samples that exit early and samples that exit late. Only after the calculation of the latest exiting sample is completed can the computation of the next batch begin. The larger the batch size, the more evident this issue becomes.

To resolve this issue, we also design an efficient acceleration method based on ELBERT, which combines the early exit mechanism artfully with batch-level parallel computing. Experimental results demonstrate that during inference,
ELBERT is able to achieve an adaptive speedup varying from $2 \times$ to $10 \times$ with negligible accuracy loss in many NLP tasks. Furthermore, by using our acceleration method, the FSA system acquires a speedup of nearly $40 \times$ without operator or CUDA level optimizations.

Our proposed datasets and the code of ELBERT (using Pytorch) are publicly released on Github, and our main contributions include the following:

- We introduce a fast and light ELBERT\textsuperscript{1} model, which has fewer parameters (18M) and uses our proposed CWB early exit mechanism to accelerate the inference. Our model outperforms existing early exit methods in terms of accuracy on various NLP tasks, including FSA, while keeping computational costs similar.
- Based on ELBERT, an efficient text processing acceleration method is developed to address the contradiction between a large input batch size and the early exit mechanism. With this method, the model can avoid the problem of computational efficiency degradation caused by samples of different difficulty exiting at different layers. As far as we know, this is the first method among open literature that can do this.
- We built a high-speed and high-accuracy FSA system, leveraging ELBERT and the proposed acceleration method, and implemented it on an Nvidia RTX 3090 GPU. With satisfactory classification accuracy, we achieved throughputs of 1503 text/s and 1107 text/s on two FSA datasets, respectively.
- We also propose a new FSA dataset labeled by our experts, which is also available in another Github project\textsuperscript{2} and evaluated in our experiments.

The rest of this paper is organized as follows. Section II gives a review of the background and related works. The ELBERT model is presented in Section III. In Section IV, we introduce the FSA system along with the ELBERT-based acceleration method on GPU and the proposed datasets. Experimental results are given in Sections V and VI. Section VII concludes this paper.

II. BACKGROUND AND RELATED WORKS

In this section, we first introduce the Transformer model and Transformer-based pre-trained language models, which form the backbone of our FSA system. We then provide an overview of related works, which are mainly distributed across two domains, BERT compression and FSA. However, our work goes beyond simply applying existing model compression methods to the FSA task. A significant advantage of our work over existing works on BERT compression is that we address the problem of computational efficiency degradation of the early exit method as the batch size increases. Additionally, unlike most related works on FSA that focus on accuracy, we give more consideration to the throughput, which has been largely ignored. While Mishev et al. [5] compared the performance of different compressed BERT models on FSA datasets, they did not discuss the speed or latency of these models. In the following subsections, we provide further details.

A. Transformer and Transformer-Based Pre-Trained Models

The Transformer model (Fig. 2) [9], invented in 2017, has achieved significant success and is widely used in the field of NLP. Compared with the recurrent layer in recurrent neural networks (RNNs), the multi-head attention layer in the Transformer can learn long-range dependencies between words in a sequence and can be trained in parallel. Recently, researchers have found that the Transformer model even outperforms convolutional neural networks (CNNs) in multiple computer vision

\textsuperscript{1}https://github.com/shakeley/ELBERT
\textsuperscript{2}https://github.com/NLP-Applications/Financial-sentiment-analysis-NLP-Transformers.git
tasks such as reading, writing, designing, and Q&A (question-answering). Big NLP models can perform numerous NLP tasks with little labeled data, achieving better performance than previous methods. BERT (Bidirectional Encoder Representations from Transformers), a representative model among these PTMs, achieved state-of-the-art performance on eleven NLP tasks [11] when it was released in 2018. Since then, many BERT-based or Transformer-based PTMs, such as RoBERTa [12], ERNIE [13], and XLNet [14], have further improved the accuracy of many NLP tasks. However, PTMs are also becoming increasingly larger, with models containing trillions of parameters being developed worldwide, including GPT-3 [15], Switch Transformer [16], and M6 [17] over the past two years.

**B. NLP Transformers Compression**

Owing to massive data used for training and extremely large numbers of trainable parameters, big NLP models can perform various tasks such as reading, writing, designing, and Q&A (questioning and answering). However, applying these models in real-world applications faces significantly challenges due to huge memory consumption and computational delay. Therefore, the model compression of NLP Transformers has also become an essential research field. Prior works in compressing BERT can be divided into two categories: structure-wise and input-wise.

1) **Structure-Wise**: Structure-wise approaches try to withdraw the trivial elements in networks. There are three common structure-wise compressing methods for BERT: weight pruning, quantization, and knowledge distilling. For weight pruning, Gordon et al. [18] involved the magnitude-based pruning, while Michel et al. [19] pruned BERT based on gradients of weights. For quantization, Q-BERT [20] employed a Hessian-based mix-precision approach to compress BERT. and Q8BERT [21] quantized BERT using symmetric linear quantization. In addition, knowledge distilling was applied by Tang et al. [22], Sun et al. [23], DistillBERT [24] and TinyBERT [25] to build lightweight BERTs. Compared with the compressed models based on these techniques, ALBERT [6] drastically reduced the number of parameters and storage consumption by parameter-sharing strategy and even outperformed BERT.

2) **Input-Wise**: Input-wise methods avoid superfluous calculations by considering the complexity of inputs. For example, BranchyNet [26] proposed an entropy-based confidence measurement, while Shallow-Deep Nets [27] solved overthinking issues via early exit mechanisms. DeeBERT [28] and TheRT [29] applied the basic early exit method to BERT. FastBERT [7] proposed a self-distilling method during fine-tuning. Unlike these works that only focus on the confidence of the classifier, our CWB mechanism also incorporates the historical trend of the classifier prediction.

It is worth mentioning that this classification method for prior works inspires ELBERT, which will be further discussed in Section III. Moreover, From another point, due to the requirements of speed and accuracy, FSA is an excellent area to study model compression of BERT, and the details are provided as follows.

**C. FSA**

Unlike general sentiment analysis, which may not effective in specific domains like finance, FSA aims to predict the reaction of financial markets to the information extracted from news, corporate announcements, or social media. The positive or negative results of FSA can be used by investors and traders to adjust their decisions in advance [5]. Previous works have shown the effectiveness of applying machine learning techniques in FSA tasks. For instance, [4] employed the Support Vector Machine (SVM) classifier to sentiment analysis of financial tweet streams, and found a correlation between the positive sentiment and the changes in stock closing prices. [30] predicted stock-market movements by applying Long short-term memory (LSTM) networks to company announcements, and showed the deep learning method is preciser than conventional approaches. However, compared to NLP Transformers, these methods are less accurate [5].

BERT or Transformer encoder layer, a commonly used text feature extractor in various AI applications, is also applied in the FSA task with promising results. FinBERT [31], a BERT-based model designed for NLP tasks in the financial domain, achieved state-of-the-art results on two FSA datasets. [5] researched numerous NLP-based methods for FSA, and concluded that NLP Transformers show superior performances compared to various sentiment analysis approaches. To extract key information from the online financial text and conduct public opinion analysis in social media, [32] proposed a sentiment analysis and key entity detection approach based on BERT. The event-driven trading strategy, an important application of FSA, utilizes the results of classifying financial texts to make buying or selling decisions.
decisions [33]. Recent studies [34], [35], [36] also utilize BERT or Transformer encoder layers to increase the text classification accuracy and profitability. These works demonstrate that BERT or Transformers can significantly improve the accuracy of FSA. However, the time consumption of these novel networks has not received sufficient attention.

As an integral component of text processing within a quantitative investment or event-driven trading system, the throughput of an FSA system is a crucial performance metric. On one hand, when a significant market-impacting event occurs, time is limited for investment decisions. Financial experts acknowledge that the market’s absorption of new information is indeterminate, which can range from minutes to hours [37]. On the other hand, in the current era of information explosion, online news and opinions are being generated exponentially through social media development. This presents unique investment opportunities, but only if we can capture the text’s temporal granularity and respond to market changes faster and more effectively than other traders [8]. Therefore, a high-performance FSA system is desired to process as much text data within a given time frame or process a fixed amount of data in less time, ensuring that investment decisions are made quickly and accurately.

Several FSA and event-driven trading studies have emphasized the significance of processing speed. However, they lack discussion on applying model compression or acceleration techniques. For example, authors of [35] highlighted the importance of quick trading after news posting for event-driven model’s profitability, but they did not discuss the speed of their Transformer-based detection model. Similarly, Mishev et al. fine-tuned lightweight BERT models for FSA and compared their accuracy, but did not measure the latency or throughput of each model.

While NLP Transformers have achieved remarkable accuracy, networks may bring higher latency and increased slippage, which prior studies have overlooked. Therefore, the question remains: Can we maintain high accuracy while boosting the performance of each model.

Several FSA and event-driven trading studies have emphasized the significance of processing speed. However, they lack discussion on applying model compression or acceleration techniques. For example, authors of [35] highlighted the importance of quick trading after news posting for event-driven model’s profitability, but they did not discuss the speed of their Transformer-based detection model. Similarly, Mishev et al. fine-tuned lightweight BERT models for FSA and compared their accuracy, but did not measure the latency or throughput of each model.

While NLP Transformers have achieved remarkable accuracy, networks may bring higher latency and increased slippage, which prior studies have overlooked. Therefore, the question remains: Can we maintain high accuracy while boosting the processing throughput of FSA? In this paper, we will explore this question using ingenious model compression techniques.

### III. ELBERT Model

In this section, we explicate our contribution to BERT model compression through the proposed CWB early exit mechanism and ELBERT model. The discussion is centered on the network architecture and computation flow of ELBERT.

#### A. Overview

By enforcing parameter sharing across all encoder layers, ALBERT dramatically reduces the number of parameters with competitive accuracy. However, this approach can not reduce computational delay. To address this limitation, we propose the ELBERT, which builds upon the ALBERT architecture, as depicted in Fig. 3. ELBERT leverages the CWB early exit mechanism to significantly enhance the average inference speed without introducing additional parameters or training overhead. By combining structure-wise and input-wise compression methods, we effectively eliminate redundant computations, enabling us to obtain a lightweight model with fewer parameters and adjustable computing time. The subsequent sub-sections will detail the training and inference procedures of ELBERT.

#### B. Training

During training, ELBERT computes the prediction losses of inputs exiting at different depths to enable the early exit mechanism for inference. Specifically, the loss is computed using the Cross-Entropy function in the $i$-th layer, denoted as $L_i$.

$$L_i = -\sum_{c \in C} \log P(\hat{y}_i = c \mid h_i).$$

Here, $c$ denotes a single class label, and $C$ represents the set of all class labels. Previous studies [7], [28] have conventionally computed the total loss $L$ by simply summing up individual layer losses $L_i$. To improve training outcomes across various loss combinations, we assign a trainable variable $t_i$ with an initial value of 4 to each layer, inspired by [38] and [39]. This allows the weights of each layer’s loss to be dynamically adjusted, enabling us to cover different cases and achieve better training results. Specifically, the weight $w_i$ of the $i$-th layer’s loss is calculated as follows:

$$w_i = \begin{cases} \sigma(t_i) & 0 < i < d - 1 \\ d - \sum_{i=1}^{d-1} \sigma(t_i) & i = d \end{cases}.$$

Note that we use $\sigma(\cdot)$ refers to sigmoid function $\sigma(t_i) = 1/(1 + \exp(-t_i))$, and the depth of ELBERT is defined as $d$. Afterwards, we can calculate the total loss $L$ with a weighted sum

$$L = \sum_{i=1}^{d} w_i \cdot L_i.$$
By this means, the cases where the input may drop out at different encoder layers can be well regarded, which is very helpful in bridging the gap between training and inference for ELBERT.

C. Inference

The CWB mechanism employed by ELBERT is a two-stage process that utilizes the intermediate state and historical trends of the classifier output to determine whether to terminate the computation prematurely. To aid in understanding this mechanism, consider the analogy of taking an exam. When a correct answer is confidently identified, further checking may be unnecessary, saving valuable time. Similarly, if several checking results indicate a clear trend towards a certain option, additional calculations may not be warranted.

In the first stage of the CWB mechanism, the model decides whether to terminate early based on whether the exit threshold has been reached. If the threshold has been reached, the computation of the input sample will be terminated early. Otherwise, the model enters the second stage.

In the second stage, the historical trend of the classifier output is further considered. If the trend stabilizes, the exit condition is met, and the computation is terminated to save time.

Specifically, during inference, the input sentence $x$ first passes through the embedding layer, producing the output $h_0$. The hidden state tensor $h_i$ then iteratively passes through the encoder.

$$
    h_i = \begin{cases} 
    \text{Encoder}(h_{i-1}) & 0 < i \leq d \\
    \text{Embedding}(x) & i = 0 
    \end{cases}
$$

Following each iteration, $h_i$ is used as input for the classifier, which generates a prediction probability distribution $p_i = \text{Classifier}(h_i)$ through the use of a fully-connected (FC) layer and softmax function for classification. Subsequently, the predicted label is then calculated as $\hat{y}_i = \text{argmax}(p_i)$.

The first stage of the CWB mechanism focuses on the classifier’s confidence or intermediate state. Given a probability distribution $p_i$, the normalized entropy is taken as the Puzzlement of the current classifier:

$$
    \text{Puzzlement}(i) = \frac{\sum_{j=1}^{C} p_i(j) \log p_i(j)}{\log(1/C)}.
$$

$C$ represents the number of labeled classes. When $\text{Puzzlement}(i) < \delta$, where $\delta$ is a user-defined threshold, the model will stop the inference in advance and take $\hat{y}_i$ as the final prediction to skip further computations. By adjusting the value of $\delta$, a faster model can be achieved at the cost of a slight reduction in accuracy, or vice versa.

The second stage of the CWB mechanism involves tracking the historical trend of the classifier output over a time window. The size $W$ of this window should be chosen appropriately, as a window that is too small may result in a less stable trend, whereas a window that is too large may prevent the model from achieving significant acceleration. To determine the optimal exit point, we propose three criteria for triggering an early exit in a time window:

- The prediction probability distribution $p_i$ is more and more biased towards a certain class in $W$ times;
- The variation range of $p_i$ is less than a set value in $W$ times;
- The predicted label $y_i$ stays the same in $W$ times.

Experimental results suggest that the first criterion outperforms the other two. Thus, in subsequent experiments, we default to using the first criterion for the second stage of the CWB mechanism. Through experimentation, we determine that the optimal value for $W$ is 8, which yielded the best performance on the validation set. We recommend using $W = 8$ when utilizing ELBERT, or experimenting with slight adjustments to this value.

Typically, the goal is to achieve sufficient confidence as early as possible. The second stage early exit is only considered when the first stage condition is not satisfied, providing comprehensive coverage for the possibility of early exit and mitigating the issue of overthinking, which can lead to the model placing undue emphasis on misleading information. It is worth noting that, by adjusting a single parameter without retraining, ELBERT can achieve different accuracy-speed tradeoffs during inference with the pre-trained model.

IV. ELBERT-BASED FSA SYSTEM

A. Overview

Fig. 4 provides a brief introduction to the FSA system utilized in this work. Before the classification stage, the textual data undergoes pre-processing to eliminate irrelevant information. We adopt a portion of the data pre-processing methodology outlined in a prior study [40], which is a common technique, and make necessary adjustments to suit our requirements. Our pre-processing procedure encompasses four steps: 1) Removal of special symbols: Texts often contain various symbols, such as full stops, quotation marks, and ellipses, which do not convey any sentiment. 2) Removal of stop words: Stop words, including conjunctions and determiners, add no meaning to a sentence. 3) Removal of non-financial content: Certain texts contain information irrelevant to finance. We also eliminate news sources and web links that relate little to financial sentiment. 4) Data augmentation: We employ back-translation [41] as a data augmentation approach in our system.

Subsequently, we select NLP Transformers as the classification model, which have been pre-trained on large corpora.
Fig. 5. Several cases to explain the advantages and the basic idea of the proposed acceleration mechanism based on ELBERT. An example is given to help understand how different samples with different latencies are parallelized in Case 6.
and fine-tuned on FSA datasets during the training stage. It is noteworthy that Transformer networks account for the majority of system delay during the inference phase. Specifically, if ELBERT is employed as the classification model in this work, we can employ a clever method to enhance inference speed, which will be discussed in greater detail in Section IV-B.

B. Inference Acceleration of ELBERT on GPU

Servers equipped with GPUs have become ubiquitous computing devices, particularly in the fields of artificial intelligence and cloud computing. They have also been widely adopted by quantitative investment institutions. Here we give an innovative and inspiring method to accelerate BERT on GPU based on the proposed ELBERT. The method combines the early exit mechanism with batch-level parallel computing, which was previously a challenging problem.

Fig. 5 illustrates the basic idea and advantages of the proposed acceleration mechanism through several cases of BERT acceleration on GPU. The figure also depicts the relationship between GPU hardware utilization and computational delay for each case on the right side of the graph. In the following subsections, we provide a detailed description of this figure.

1) Case 1: In this scenario, no additional acceleration technique is required as the batch size is set to 1. The input sentence is passed through an embedding layer, resulting in an output tensor with a shape of (seq_len, hidden_state), where “seq_len” denotes the maximum sequence length across all input sentences, and “hidden_state” means the model’s dimension (i.e., \( d_{model} \) in [9]). It is noteworthy that the input and output tensors of a Transformer encoder share the same shape, so encoders in Case 1 will produce inputs and outputs with a shape of (seq_len, hidden_state).

2) Case 2: In Case 2, the calculation in an intermediate layer of BERT terminates in advance once the exit condition is satisfied. This approach avoids redundant operations and accelerates the computation. Compared with Case 1, the speedup of Case 2 depends on how many layers the network skips. However, this strategy does not enhance the GPU hardware utilization, which is the most widely utilized hardware coprocessor for deep learning.

3) Case 3: Case 3 introduces another commonly used technique to accelerate neural network inference. When using a GPU to accelerate neural network inference (which is also true for training), setting the batch size to 1 may result in low utilization of computation or memory resources. To improve parallelism and reduce overall latency, increasing the batch size is a straightforward solution. If we enlarge the batch size under a specific limit, the throughput of the inference system will be enlarged by a nearly equal proportion. However, unlike early exit, this approach does not reduce the computational load of the network itself.

4) Case 4: Compared to model compression techniques like quantization and pruning, which often require dedicated hardware to accelerate the network, the early exit is device-agnostic. However, it is not well-suited for increasing the batch size, as demonstrated in Case 4. In this scenario, different samples within a batch may exit at different network layers, resulting in the latency of the entire network being determined by the slowest sample in every batch. This is commonly described as the “Wooden Barrel Effect”: The amount of water a barrel can hold depends on the shortest plank. Hence, early exit can only achieve better speed-up when the batch size is set to one. As a baseline model in the experimental section, although FastBERT can also support the early exit mechanism while allowing the input batch size to be greater than one, it is in line with Case 4. FastBERT also uses the self-distillation technique to compress the model, but it cannot share the same parameters among all the encoder layers as ELBERT does. Therefore, FastBERT cannot reach the effects achieved by Case 6, which will be introduced below.

5) Case 5 and Case 6: The ELBERT model is a promising solution to the incompatibility between early exit and increasing batch size. This is achieved through the encoder-level parameter sharing mechanism, which allows the GPU to load only one encoder layer, and the computation between the embedding layer and the classifier becomes a repetitive iteration process. For example, although input sentence A and input sentence B are not sent into the network simultaneously (maybe A is starting its fifth iteration, but B has just passed through the embedding layer), they can still be computed in a single batch. Through input sequence adjustment, we can fill up every input tensor of the encoder during most of the computation (except for a few iterations near the end) to ensure that its shape is always \((N, seq_len, hidden_state)\), ensuring high GPU utilization and further improving text processing speed.

Algorithm 1 provides a detailed computational flow of the acceleration method. The input tensor of the encoder is denoted as \( \hat{S} \), and its shape is \((N, seq_len, hidden_state)\). The variable \( \hat{F}_i \) marks whether the \( i \)-th sample in \( \hat{S} \) needs to be computed, while \( \hat{L}_i \) saves how many times the \( i \)-th sample has passed through the encoder layer. The method consists of two stages, which are described in a while loop in Algorithm 1 from line 6 to line 23. In the first stage, \( \hat{S} \) is filled up, and utilized as the input for the second stage. The while loop processes all samples from the input dataset to the encoder layer, and if the network has not completed all computations, the last part of the code (starting from line 26) completes the task. The encoder primarily causes the latency of the entire model, but the stage of data filling also requires careful handling to avoid time-consuming serial operations when the batch size is increased. For example, the embedding layer (line 9) should process input sentences in parallel to speed up the calculation as much as possible.

There are several important points to consider with regard to this acceleration approach. Firstly, it primarily focuses on high-level optimization, making it highly portable. As the internal calculation of each encoder is not modified, many other techniques can be compatible with this method to further improve its performance, such as model distillation, low-bit quantization, or underlying operator optimization. Secondly, this approach can be extended to other BERT-like models that support early exit and parameter-sharing strategies. However, it is highly unlikely to be applied to models like CNNs due to their different input and output tensor shapes for each layer, making the
Algorithm 1 ELBERT-Based Acceleration Method

1: Input: The input dataset
2: \( S \leftarrow \) An empty tensor with shape as \( (N, \text{seq_len, hidden_state}) \)
3: \( F_{1,2,\ldots,N} \leftarrow \) [False, False, ... , False]
4: \( L_{1,2,\ldots,N} \leftarrow \) [0, 0, ... , 0]
5: Return List \( \leftarrow \) [0, 1, ... , N-1]
6: while Input dataset is not empty do
7: \( k \leftarrow \) len(Return List)
8: Input Sentence \( \leftarrow \) the next \( k \) samples from the input dataset
9: add Embedding_Layer(Input Sentence) to \( \hat{S} \)
10: \( F_{1,2,\ldots,N} \leftarrow \) [True, True, ... , True]
11: Return List \( \leftarrow \) empty
12: while \( F_{1,2,\ldots,N} \neq \) [False, False, ... , False] do
13: \( \hat{S} \leftarrow \) Encoder(\( S \))
14: \( L_{1,2,\ldots,N} \leftarrow L_{1,2,\ldots,N} + 1 \)
15: for \( i = 0, 1, \ldots, N - 1 \) do
16: if \( S_i \) can exit OR \( L_i = d \) then
17: \( F_i \leftarrow \) False
18: add \( i \) to Return List
19: \( L_i \leftarrow 0 \)
20: end if
21: end for
22: end while
23: while \( F_{1,2,\ldots,N} \neq \) [False, False, ... , False] do
24: for \( i = 0, 1, \ldots, N - 1 \) do
25: if \( F_i \) = False then
26: \( \hat{S}_i \leftarrow \) An empty tensor with shape as \( (1, \text{seq_len, hidden_state}) \)
27: end if
28: end for
29: \( \hat{S} \leftarrow \) Encoder(\( S \))
30: \( L_{1,2,\ldots,N} \leftarrow L_{1,2,\ldots,N} + 1 \)
31: for \( i = 0, 1, \ldots, N - 1 \) do
32: if \( S_i \) can exit OR \( L_i = d \) then
33: \( F_i \leftarrow \) False
34: \( L_i \leftarrow 0 \)
35: end if
36: end for
37: end while
38: return The classifier results of all the input samples
39:

C. The Proposed SK Dataset

A lack of labeled datasets represents another challenge for financial sentiment analysis (FSA) [5]. To address this issue and provide more data for experimentation, we constructed a new FSA dataset derived from the Seeking Alpha website. Seeking Alpha is a widely popular investment research website featuring freelance contributors’ discussions of the financial markets. Founded in 2004, the site has garnered several million registered users. Thousands of industry analysts contribute content to the site, posting commentary and analyses of companies and equities globally. Our expert manually labeled a subset of this content to create the new FSA dataset.

The SK dataset contains the hottest financial news on SeekingAlpha from March 2005 to April 2021. The news covers topics like company performance, international situation, and national policies. Based on their probable impact on financial markets, we classified all news into Positive, Negative, and Neutral categories. To avoid imbalanced samples for training, we selected 1,452 news items. The examples and distribution can be seen in Tables I and II.

For more details on this dataset, please refer to our GitHub project.

V. EXPERIMENT RESULTS ON GENERAL NLP DATASETS

This section evaluates the effectiveness of the CWB early exit mechanism and the ELBERT model. The experimental results, shown as accuracy versus computational cost curves, demonstrate that ELBERT achieves excellent inference acceleration and outperforms other early exit methods used to accelerate BERT. The computational cost is a relative value, with a cost of 1 representing no early exit. It is not specific to any computing device.

To comprehend the principles of the CWB early exit mechanism, we also visualize ELBERT’s decision-making process. ELBERT’s ability to improve text processing speed on GPUs and the performance of the overall FSA system will be demonstrated in the next section.

A. Baselines

To demonstrate the advantages of the CWB early exit mechanism, here we choose three baselines:

1. https://seekingalpha.com
Fig. 6. The accuracy-speed tradeoff of ELBERT on different datasets. The computation cost represents the normalized ratio of the original computation. The rightmost point of each curve denotes the original model without early exit.

- Original model: ALBERT-large (depth=24) is selected in fine-tuning and inference.
- Plain compression: We evaluate several smaller-sized models based on ALBERT-large in fine-tuning and inference. Notice that ALBERT has only one encoder, making it possible to set depth manually in inference to achieve different speeds.
- Early exit approach: For a fair comparison, the early exit methods in DeeBERT and FastBERT are all applied to ALBERT-large.

B. Datasets

In this section, to test the generalization capability of ELBERT, commonly used GLUE benchmark, IMDB, and AG-news are evaluated in the experiment. These datasets contain diverse NLP tasks such as Natural Language Inference, News Classification, and Sentiment Analysis.

C. Training and Inference Settings

1) Training: The corresponding hyperparameters are kept the same as for GLUE for a fair comparison. For AG-news and IMDB, we employ a learning rate of 3e-5 and a batch size of 32.

2) Inference: To focus on the comparison between different early exit mechanisms, we set the batch size of inference to 1, following prior work. The experiments in this section are all done on an NVIDIA 2080Ti GPU.

D. Performance of ELBERT

1) Accelerating Inference Flexibly: ELBERT’s performance is evaluated on the datasets mentioned above. The median of 5 runs is reported in Figs. 6 and 7, where the y-axis is accuracy and the x-axis is compute-ratio, the normalized ratio of computation based on FLOPs of samples. The curves are drawn by interpolating several points that correspond to different δ values, varying from 0.1 to 1.0 with a step size of 0.1 in the first stage of the early exit. ELBERT achieves at least two times inference speedup for all datasets while maintaining or even improving accuracy. The inference acceleration ratio can reach up to ten times with a tolerable accuracy degradation. These results demonstrate ELBERT’s superiority in accelerating inference.

2) Task-Related Trends: It is worth mentioning that the trends of the curves in Fig. 6 are quite different. For example, in the news classification task where ELBERT performs best, the accuracy loss at the lowest computation cost remains very little. In other tasks like sentiment analysis (SST-2, IMDB) and natural language inference (QNLI, RTE), the curves drop more steeply as the computational cost decreases. This suggests that different tasks may have different internal attributes and acceleration difficulties.

3) Flexible and Better Accuracy-Speed Tradeoffs: Comparison results are exhibited in Fig. 7, where the red star-shaped points represent different models obtained through plain compression. Our first observation is that ELBERT significantly outperforms plain compression models. Furthermore, compared to other early-exit methods, ELBERT acquires higher accuracy than DeeBERT and FastBERT under the same computational cost, which indicates ELBERT’s notable advantages over other approaches.

E. Visualization of CWB

We have seen ELBERT’s outstanding performance in accelerating inference. To visualize the CWB decision-making process, we modify BertViz, a tool for visualizing attention in Transformer models. The attention scores of each layer are utilized to get the cumulative attention scores, allowing us to see the attention relationships between tokens clearly as the input sample passes through ELBERT’s layers. Since ELBERT only takes the [cls] token as the representation of one sentence to
Fig. 8. A simple case. The early exit is triggered when the attention to specific word (hampered) exceeds a certain limit.

Fig. 9. A hard case. The early exit is triggered in time after the commendatory word (benign) is well noticed, avoiding subsequently overthinking about unrelated negation (rarely).

VI. EXPERIMENTS RESULTS OF OUR FSA SYSTEM

A. The Relationship Between Inference Batch Size and Speed on GPU

Before presenting the results of the FSA task, we conduct some experiments to demonstrate that the inference speed of neural networks can be improved by increasing the batch size of the input, as mentioned in Section IV-B. Some results are shown in Fig. 10, where the SK dataset is used as the input example, and the ALBERT-large model is deployed on an Nvidia RTX3090 GPU. In this case, by increasing the batch size, the throughput can be increased from 38 texts/second to 240 texts/second.

We can see that as the batch size expands within certain limits (around 6 or 7), the latency for each batch increases very slowly, causing the throughput of the entire inference system to increase almost linearly with the batch size. When the batch size increase exceeds this limit, the hardware utilization rate reaches a very high value. At this time, increasing the batch size cannot improve the throughput rate as easily. This fact is also consistent with the principle we mentioned in Fig. 5: The reason we can increase the batch size to increase speed is that it improves hardware utilization.

In short, the throughput can reach an upper limit by making the batch size larger. Therefore, in the following experiments, we only distinguish between a batch size of one and a sufficiently large batch size to maximize both GPU utilization and processing speed, and the latter is denoted as “batch size=N”.

B. Experiment Settings of ELBERT-Based FSA System

1) Baseline Models: FastBERT is chosen as the primary comparison model due to it is one of the latest lightweight BERT models, but also because of its publicly available source code, which supports input batch size greater than one. This feature is rare among publicly available models, particularly those that use early exit methods, which are typically restricted
to the batch size equal to one. Additionally, the DeeBERT model is selected, which supports only a batch size of one. (In the case of “batch size=N” in Fig. 12, DeeBERT’s performance is identical to that of “batch size=1”.) The BERT-large and ALBERT-large models are also fine-tuned on our FSA datasets for comparison. To compare network performance in the overall system, we utilize the entire FastBERT and DeeBERT models, rather than only their early exit approaches, as compared in Section V. These two lightweight BERT models, along with ELBERT, are all compressed from BERT-large and are used in the following experiments.

2) Our Model: We employ the ELBERT-based acceleration technique, as outlined in Algorithm 1, to enhance the performance of the FSA task. In the inference phase, batch sizes from 1 to 32 are tested, and the highest throughput is chosen as the case of “batch size=N”, as previously stated. All the inference experiments are conducted on a server equipped with an Nvidia RTX 3090 GPU.

3) Datasets: Our FSA system is fine-tuned and tested on both the Financial Phrase-Bank (FPB) dataset [45] and the proposed SK dataset. The FPB dataset consists of 4845 financial texts which have been annotated by financial experts. We partition the dataset such that 80% of the sentences are reserved for training, while the remaining 20% are used for validation purposes.

4) Fine-Tune: All of these training hyperparameters are adjusted to the appropriate value. Specifically, we set the learning rate to 2e-5 for FastBERT and 3e-5 for other models. The training batch size for all models is set to 32. All these models take ten epochs to finish the fine-tuning, and the FastBERT requires another ten epochs for self-distillation.

C. Results of ELBERT for FSA Task

We utilize ELBERT and the proposed ELBERT-based acceleration mechanism to expedite the FSA task. Here we present and analyze the results through horizontal and longitudinal contrast. Notably, we establish an “Accuracy with tolerable loss” threshold for each dataset depicted in Fig. 12. The figure reveals that a smaller accuracy loss above the threshold corresponds to a greater speed gain, while the accuracy loss becomes unmanageable below the threshold. We employ this threshold as a yardstick for the comparisons presented in Tables III and IV.

The results depicted in Fig. 12 demonstrate that ELBERT significantly outperforms all the baseline models in terms of accuracy and throughput for the FSA task. Notably, the original BERT and ALBERT models lack an early exit mechanism, thus their performance is represented in the graph as individual points rather than curves connected by multiple points. By decreasing the exit threshold, ELBERT can easily achieve a much higher speed than these two baseline models while maintaining comparable accuracy. Additionally, although the DeeBERT model can be accelerated by an early-exit mechanism, it only supports a batch size of one (as reflected in Fig. 12), where it performs inferiorly to ELBERT and FastBERT.

The best-performing baseline model, FastBERT, employs an early exit mechanism and supports batch sizes larger than one. However, even in this scenario, ELBERT still outperforms FastBERT, as demonstrated by the model accuracy-throughput curves in Fig. 12. Specifically, when the batch size is one, the curves of ELBERT are closer to the top right corner, indicating that ELBERT achieves faster processing speeds while maintaining the same level of accuracy as FastBERT, or higher accuracy with the same speed. Moreover, as the batch size increases, the performance gap between ELBERT and FastBERT widens, with our method being nearly twice as fast as FastBERT while maintaining sufficient accuracy. The intersections of the model accuracy-throughput curves and the two “Accuracy with tolerable loss” lines represent the best trade-off between inference speed and acceptable accuracy loss. The horizontal coordinates (throughput rates) of these intersections of the curves (ELBERT and FastBERT) are organized in Table III. As evident from the table, when the batch size is one, ELBERT is only 1.4~1.5 times faster than FastBERT. Furthermore, when the batch size is N, ELBERT is 1.8~1.9 times faster than FastBERT, highlighting the efficacy of our acceleration method. Notably, the encoder layers of FastBERT have different parameters, rendering it incapable of utilizing the acceleration method proposed in Section IV.

The results of ELBERT demonstrate that our ELBERT-based acceleration method can effectively leverage the early exit mechanism and higher parallelism to achieve improved performance. The speedup achieved by our approach is presented in Table IV. Specifically, in experiments using the FPB dataset, our approach achieves a speedup of 3.13× with an acceptable accuracy loss through the early exit mechanism. By increasing the batch size, another 12× speedup is achieved. Similar results are obtained when using the SK dataset, with speedups of 5.26× and 7.94× achieved through the early exit mechanism and increased batch size, respectively. These results attest to
Fig. 11. Comparison between the ELBERT and baseline models applied in FSA tasks. In “batch size=N” scenario, ELBERT outperforms FastBERT, the best-performing baseline model, by a significant margin, thereby showing the improvement brought by our acceleration method. The points above the dotted lines have acceptable accuracy loss, and “batch size=N” (1 < N ≤ 32) means the batch size is large enough to enable the processing speed to reach an upper limit. These results demonstrate that our method beats all the baseline models, including the original BERT and other light BERTs.

D. Ablation Study

To further illustrate the effectiveness of this acceleration method, we also provide the results of the ablation experiments, as shown in Fig. 12. First, we do not use any optimization scheme, but simply increase the batch size from the original ELBERT (Case 4 in Fig. 5), and we name the accuracy-throughput curve measured in this setting as ELBERT-original. Second, ELBERT-intermediate uses the proposed acceleration method but does not optimize the data filling. Finally, by utilizing the acceleration method along with optimizing the data-filling process, the curves marked ELBERT-best are the best performance of ELBERT, and in fact, they are the same as the results of ELBERT shown in Image 11 with batch size=N.

With the data provided in Fig. 12, we can draw some further conclusions. First of all, our carefully designed acceleration method has a significant effect on the throughput compared to simply increasing the batch size. With acceptable accuracy (0.907 on the FPB dataset, and 0.860 on the SK dataset), ELBERT-original can only achieve throughputs of 500 text/s on the FPB dataset and 333 text/s on the SK dataset, which are only 33% and 30% of ELBERT-best, respectively. With the same accuracy, although ELBERT-intermediate also ensures that the input tensor during each encoder computation is as free of invalid data as possible, its throughput is also only 54% and 62% of ELBERT-best due to the lack of optimization for data filling. This result reflects that the optimization of data filling is also very important.

Thus, the results of the ablation experiments also demonstrate the effectiveness of the proposed ELBERT-based acceleration method in throughput improvement.

VII. CONCLUSION

This paper introduces a compressed BERT model and demonstrates its great potential in NLP tasks desiring both high throughput and high accuracy, such as FSA.

Firstly, the ELBERT and a CWB early exit mechanism are proposed. The ELBERT has fewer parameters (18M) and faster inference speed (2~10× speedup on various datasets). The CWB mechanism also outperforms existing early exit methods used for accelerating BERT on many NLP tasks. Based on the ELBERT, a novel acceleration method is developed to creatively solve the incompatibility between early exit and high parallelism on GPUs. This method also helps us to build a fast and high-accuracy FSA system that can label more text data at the same time or finish processing the same input samples with lower latency. ELBERT achieves an accuracy of 0.907 with a throughput of 1503 text/s on the FPB dataset, and 0.860 accuracy with a throughput of 1107 text/s on the SK dataset. Compared with other compressed BERT models, ELBERT outperforms them in terms of accuracy and speed in the FSA task.

Since the ELBERT model and the acceleration method are inspiring and portable, they could be applied to more NLP applications in the future. We also intend to improve our FSA system for quantitative or event-driven investment strategies.

REFERENCES

[1] K. Xie, S. Lu, M. Wang, and Z. Wang, “ELBERT: Fast ALBERT with confidence-window based early exit,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP). Piscataway, NJ, USA: IEEE, 2021, pp. 7713–7717.

[2] A. K. Davis, W. Ge, D. Matsumoto, and J. L. Zhang, “The effect of manager-specific optimism on the tone of earnings conference calls,” Rev. Accounting Stud., vol. 20, no. 2, pp. 639–673, 2015.
W. Zhang and S. Skiena, “Trading strategies to exploit blog and news sentiment,” in Proc. 4th Int. AAAI Conf. Web Social Media, 2010, pp. 375–378.

J. Smalović, M. Grčar, N. Lavrač, and M. Žnidarič, “Stream-based active learning for sentiment analysis in the financial domain,” Inf. Sci., vol. 285, pp. 181–203, 2014.

K. Mishev, A. Gjorgjejikj, I. Vodenska, L. T. Chitkushev, and D. Trajanov, “Evaluation of sentiment analysis in finance: From lexicons to transformers,” IEEE Access, vol. 8, pp. 131662–131682, 2020.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “ALBERT: A lite BERT for self-supervised learning of language representation,” in Proc. ICLR, 2019. [Online]. Available: https://openreview.net/forum?id=H1eA7AEtV5

W. Liu, P. Zhou, Z. Wang, Z. Zhao, H. Deng, and Q. Ju, “FastBERT: A self-distilling BERT with adaptive inference time,” in Proc. 58th Ann. Meeting Assoc. Comput. Ling., 2020, pp. 6035–6044.

R. Savchny, A. Wadhwa, S. Agarwal, and R. Shah, “Fast: Financial news and tweet based time aware network for stock trading,” in Proc. 16th Conf. Eur. Assoc. Comput. Ling., 2021, pp. 2164–2175.

A. Vaswani et al., “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 5998–6008.

Z. Liu et al., “Swin Transformer: Hierarchical vision transformer using shifted windows,” in Proc. IEEE/CVF Int. Conf. Comput. Vision, 2021, pp. 10019–10029.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. NAACL, 2019, pp. 4171–4186.

Y. Liu et al., “RobERTa: A robustly optimized BERT pretraining approach,” 2019, arXiv:1907.1192.

Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, “ERNIE: Enhanced language representation with informative entities,” 2019, arXiv:1905.07129.

Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, “XLNet: Generalized autoregressive pretraining for language understanding,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 5753–5763.

L. Floridi and M. Chiriatti, “GPT-3: Its nature, scope, limits, and consequences,” Minds Mach., vol. 30, no. 4, pp. 681–694, 2020.

W. Fedus, B. Zoph, and N. Shazeer, “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity,” 2021, arXiv:2101.03961.

J. Lin et al., “M6: Multi-modality-to-multi-modality multi-task transformer for unified pretraining,” in Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Min., 2021, pp. 3251–3261.

M. A. Gordon, K. Duh, and N. Andrews, “Compressing BERT: Studying the effects of weight pruning on transfer learning,” 2020, arXiv:2002.08307.

P. Michel, O. Levy, and G. Neubig, “Are sixteen heads really better than one?” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 14041–14042.

S. Shen et al., “Q-BERT: Hessian based ultra low precision quantization of BERT,” in Proc. AAAI, 2020, pp. 8815–8821.

O. Zafrir, G. Boudoukh, P. Izsa k, and M. Wasserblat, “Q8BERT: Quantized 8bit BERT,” 2019, arXiv:1910.06188.

R. Tang, Y. Lu, L. Liu, L. Mou, O. Vechtomova, and J. Lin, “Distilling task-specific knowledge from BERT into simple neural networks,” 2019, arXiv:1903.12136.

S. Sun, Y. Cheng, Z. Gan, and J. Liu, “Patient knowledge distillation for BERT model compression,” in Proc. EMNLP-IJCNLP, 2019, pp. 4314–4323.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter,” 2019, arXiv:1910.01108.

X. Jiao et al., “TinyBERT: Distilling BERT for natural language understanding,” 2019, arXiv:1909.10351.

S. Teerapittayanon, B. McDaniel, and H. T. Kung, “BranchyNet: Fast inference via early exiting from deep neural networks,” in Proc. ICLR, 2016, pp. 2464–2469.

Y. Kaya, S. Hong, and T. Dumitras, “Shallow-deep networks: Understanding and mitigating network overthinking,” in Proc. ICML, PMLR, 2019, pp. 3301–3310.

J. Xin, R. Tang, J. Lee, Y. Yu, and J. Lin, “DeeBERT: Dynamic early exiting for accelerating bert inference,” 2020, arXiv:2004.12993.

R. Schwartz, G. Stanovsky, S. Swayamdipta, J. Dodge, and N. A. Smith, “The right tool for the job: Matching model and instance complexities,” in Proc. ACL, 2020, pp. 6640–6651.
Keli Xie received the B.S. degree in integrated circuit design and integrated system from Nanjing University, Nanjing, China. He is currently pursuing the master's degree in integrated circuit design at Nanjing University. His current research interests include efficient hardware for AI, model compression, and text summarization, especially dialogue summarization.

Jun Lin (Senior Member, IEEE) received the B.S. degree in physics, M.S. degree in microelectronics from Nanjing University, Nanjing, China, in 2007 and 2010, respectively, and the Ph.D. degree in electrical engineering from Lehigh University, Bethlehem, in 2015. From 2010 to 2011, he was an ASIC Design Engineer with AMD. During summer 2013, he was an Intern with Qualcomm Research, Bridgewater, NJ, USA. In June 2015, he joined the School of Electronic Science and Engineering at Nanjing University, where he is an Associate Professor. He was a member of the Design and Implementation of Signal Processing Systems (DISPS) Technical Committee of the IEEE Signal Processing Society. His current research interests include low-power high-speed VLSI design for digital signal processing and deep learning, hardware acceleration for big data processing, and emerging computer architectures. He has published over 200 technical papers with multiple best paper awards received from the IEEE Technical Societies, among which is the VLSI Transactions Best Paper Award of 2007. He is the Editor of VLSI (InTech, 2010) and held more than 20 U.S. and China patents. In the current record, he has had many papers ranking among top 25 most (annually) downloaded manuscripts in IEEE TRANSACTIONS ON VLSI SYSTEMS. In the past, he has served as an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I: REGULAR PAPERS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS, and IEEE TRANSACTIONS ON VLSI SYSTEMS for many terms. He has also served as TPC member and various chairs for tens of international conferences. Moreover, he has contributed significantly to the industrial standards. So far, his technical proposals have been adopted by more than fifteen international networking standards. In 2015, he was elevated to the Fellow of IEEE for contributions to VLSI design and implementation of FEC coding. His current research interests are in the area of optimized VLSI design for digital communications and deep learning.

Zhongfeng Wang (Fellow, IEEE) received the B.E. and M.S. degrees from the Department of Automation at Tsinghua University, Beijing, China, in 1988 and 1990, respectively, and the Ph.D. degree from the University of Minnesota, Minneapolis, in 2000. He has been working for Nanjing University, China, as a Distinguished Professor since 2016. Previously, he worked for Broadcom Corporation, California, from 2007 to 2016, as a leading VLSI architect. Before that, he worked for Oregon State University and National Semiconductor Corporation. He is a world-recognized expert in low-power high-speed VLSI design for Signal Processing Systems. He has published over 200 technical papers with multiple best paper awards received from the IEEE Technical Societies, among which is the VLSI Transactions Best Paper Award of 2007. He is the Editor of VLSI (InTech, 2010) and held more than 20 U.S. and China patents. In the current record, he has had many papers ranking among top 25 most (annually) downloaded manuscripts in IEEE TRANSACTIONS ON VLSI SYSTEMS. In the past, he has served as an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I: REGULAR PAPERS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS, and IEEE TRANSACTIONS ON VLSI SYSTEMS for many terms. He has also served as TPC member and various chairs for tens of international conferences. Moreover, he has contributed significantly to the industrial standards. So far, his technical proposals have been adopted by more than fifteen international networking standards. In 2015, he was elevated to the Fellow of IEEE for contributions to VLSI design and implementation of FEC coding. His current research interests are in the area of optimized VLSI design for digital communications and deep learning.