Knowledge Distillation Method for Compressing Large Language Model of Power Risk Identification and Improving Deployment Efficiency

Siwu Yu, Yumin He, Guobang Ban, Jintong Ma, Guanghui Xi, Lingwen Meng, Shasha Luo, Siqi Guo

Electric Power Research Institute of Guizhou Power Grid Co, Guiyang 550002, Guizhou, China

Corresponding author: Siwu Yu (e-mail: swyu2012@163.com)

ABSTRACT Large language models demonstrate high precision in power-system risk identification; however, their massive parameter counts and high resource consumption hinder real-time deployment on resource-constrained edge devices used in electromagnetic sensing systems, wearable monitoring terminals, and compact power-monitoring devices. Achieving low-latency processing in distributed sensing structures and edge-based electromagnetic monitoring devices requires a significant reduction in computational overhead to ensure immediate detection of electrical hazards, abnormal equipment states, and potential power-system risks. This paper proposes a lightweight compression method based on hierarchical supervised knowledge distillation. Experiments show that, after applying the proposed knowledge distillation method, the inference latency is reduced from 235 ms to a minimum of 26 ms, which is better than DistilBERT's 35 ms. The number of student model parameters is reduced to 4.3% of the teacher model, namely 14.5M versus 340M, while the classification accuracy reaches 89.4%, close to the teacher model's 92.7%. The F1 score for the equipment failure category reaches 90.3%, verifying the efficiency and practicality of the lightweight model in resource-constrained scenarios involving power-risk identification, electromagnetic sensing, and edge-based intelligent monitoring.

INDEX TERMS large language model, knowledge distillation, power risk identification, electromagnetic sensing, edge deployment

I. INTRODUCTION

THE current operation of power systems is highly complex and dynamic, where the frequent degradation of insulation degradation, electromagnetic interference, and equipment failures necessitate advanced intelligence and real-time performance in risk identification systems. The integration of smart sensing fibers into these volatile environments is essential for the immediate detection of communication anomalies and dispatching errors, ensuring the overall stability of the power infrastructure. The application of large language models based on natural language processing for processing unstructured data such as accident logs, inspection records, and failure report texts [1–3] has made them key tools in advancing intelligent power risk identification technology. In recent years, Transformer architecture-driven pre-trained language models such as Bidirectional Encoder Representations from Transformers (BERT), RoBERTa, Text-to-Text Transfer Transformer (T5), and Generative Pre-trained Transformer (GPT) have been widely used in accident classification, risk warning, and anomaly detection tasks in power scenarios, achieving significant breakthroughs in text understanding precision [4,5]. The goal of this paper is to develop an efficient knowledge distillation framework to compress large language models for power risk identification, specifically to facilitate high-precision edge deployment on compact power-monitoring terminals, electromagnetic sensing devices, and distributed intelligent monitoring systems. By utilizing a hierarchical supervision mechanism and quantitative optimization, this approach bridges the technical gap between complex model architectures and the real-time monitoring requirements of edge-based power safety equipment. For training, a soft label and intermediate layer feature collaborative distillation mechanism are adopted, while for deployment, a quantization and accelerated inference optimization framework is combined to build an efficient model system suitable for edge scenarios. The constructed method has good scalability and platform compatibility, providing low-latency, high-reliability, and deployable technical support for services such as automatic identification of substation faults and risk

warning on dispatching platforms, thereby promoting the implementation of smart grid semantic analysis systems in engineering applications.

II. RELATED WORK

Focusing on the task of power risk identification, different researchers have tried to alleviate the deployment burden through methods such as model structure optimization, quantization compression, and sparse coding [6]. Existing work has focused on static pruning, activation sparsity guidance, weight sharing, and similar strategies to balance accuracy and model complexity [7,8]. Some studies introduce low-rank decomposition mechanisms in Transformer models to reduce parameter redundancy; others introduce mixed-precision calculations, combining 32-bit floating point (FP32) with 16-bit floating point (FP16) to reduce storage and compute power consumption. Additionally, some works attempt to replace attention calculations with sparse

attention mechanisms to reduce spatiotemporal complexity [9, 10]. Although these methods have improved the feasibility of model deployment to a certain extent, due to the failure to fully utilize the context modeling capabilities of the original large model, the recognition accuracy is often significantly reduced during the compression process. In particular, when it comes to high-dimensional semantic concept discrimination in power scenarios, the compressed model performs poorly in terms of robustness and generalization [11,12]. Therefore, there is an urgent need to build a lightweight model that can effectively retain the knowledge of the original large model and adapt to the low-latency and high-availability requirements of power risk identification tasks.

III. MODEL LIGHTWEIGHT PRACTICE

A. TEACHER MODEL DOMAIN TUNING

1) Data Preprocessing and Domain Corpus Construction

The teacher model used in this study is RoBERTa-large, which has a pre-trained parameter size of 340M and is built based on a dynamic masked language modeling architecture. To achieve effective transfer in the power risk identification scenario, it is necessary to first build a high-quality power semantic corpus. The training data is selected from accident reports, inspection logs, and defect reports publicly released by the State Grid and China Southern Power Grid. After desensitization and reconstruction, 30,241 valid texts are obtained, all of which are samples with structured tags and unstructured descriptions [13,14]. The text length distribution is controlled between 64 and 512, with an average of 263. The unified character encoding is Unicode Transformation Format-8 (UTF-8) during data cleaning. Special characters and duplicates are deleted, and traditional and simplified Chinese are merged. Regular expressions are used to standardize time, equipment codes, and bit number information to reduce model input ambiguity.

Subsequently, the roberta-base-tokenizer is used to perform byte pair encoding (BPE) tokenization on the data. Manual word list enhancement is performed for special terms, and token segments containing highfrequency industry terms are added, such as "on-load voltage regulation fault" and "ring main unit tripping." To enhance the model's memory for long texts, the training samples are bucketed by token length and batched to ensure that the batch dimension of each training input is [batch_size=16, seq_len$\leq$512]. The total training sample size is 30,241, and the label dimension is six types of risk classification tasks. Given the significant imbalance in the number of samples across risk categories in the training set (e.g., 1,245 cases of "device malfunction" vs. only 215 cases of "security vulnerability"), a class-weighted strategy is introduced during the fine-tuning and distillation stages. Specifically, in the cross-entropy loss function, each category is assigned a weight inversely proportional to its frequency. Furthermore, during soft-label distillation, a 1.3$\times$ amplification factor is applied to the Kullback–Leibler divergence (KL) divergence loss for minority class samples to enhance the student model's learning of low-frequency but high-risk events.

2) Fine-Tuning Training Configuration and Strategy Design

In domain fine-tuning, a supervised learning strategy is adopted, with cross-entropy as the main loss function, and the goal is to maximize the classification and discrimination ability of the model in domain tasks. The training platform uses an 8$\times$NVIDIA A100 (40 GB) GPU parallel framework. The torch.nn.DataParallel module is used to load data in a distributed manner. The time required for a single epoch of training is controlled at about 52 minutes. The total number of model training epochs is set to 50. The initial learning rate is 2e-5. The optimizer selected is AdamW, with β parameters set to 0.9 and 0.999, respectively. The gradient clipping threshold is set to 1.0 to prevent convergence failure caused by large fluctuations. The learning rate scheduling strategy is WarmupLinear, where the first 10% of steps are the warmup phase, and the rest are linearly decayed. Each epoch performs a precision evaluation on the dev set (with verification samples accounting for about 10%) to dynamically monitor training stability and overfitting risks. Despite 50 training epochs, the model does not exhibit overfitting thanks to multi-strategy regularization. In addition to the aforementioned MLM-assisted tasks, Dropout (rate = 0.3), weight decay (0.01), and an early stopping mechanism (patience = 7) are introduced. Furthermore, since power texts have highly structured features (such as descriptions of fixed fault modes), a 30k-scale corpus is sufficient to support RoBERTa-large domain adaptation, which differs from the requirement of millions of data points for general natural language processing (NLP) tasks.

To enhance the model's ability to represent the semantics of the power industry, a subtask auxiliary mechanism is used during training. The masked language model (MLM) auxiliary loss branch randomly masks 15% of the tokens

in the input text. The prediction target is the original token index, and it is trained in parallel with the main task, sharing the underlying encoder but using independent gradient backpropagation. This strategy improves the model's perception of industry-specific terms and enhances convergence stability. The horizontal axis in Figure 1 is the training epoch (1–50). The left vertical axis shows that the cross-entropy loss drops from about 1.18 in the 1st epoch to about 0.11 in the 50th epoch, and the MLM loss decreases from about 0.76 to about 0.05. The right vertical axis shows that the validation accuracy increases from about 61% in the 1st epoch to about 95% in the 50th epoch. The simultaneous decrease in cross-entropy and MLM loss indicates that the main and auxiliary tasks converge collaboratively. The continuous increase in validation accuracy verifies the efficient generalization ability of power risk classification with a lightweight structure.

3) Model Output and Soft Label Extraction Preparation

After training, the teacher model retains all weight parameters and intermediate layer feature outputs. To meet the input requirements of the subsequent distillation stage, the samples in the training set are inferred again, and the softmax layer probability distribution is output as a soft label. During inference, the temperature parameter is set to T = 2 to expand the output distribution information density and enhance the student model's ability to distinguish between category differences. For each sample, the hidden states of the 6th and 10th layer Transformer modules (with a dimension of [B, L, 1024]) are retained as the source target representation for distillation intermediate feature alignment [15]. These output feature data are stored in .pt file format and cached in batches to GPU video memory or disk to improve the efficiency of subsequent student model training. After the above operations are completed, the RoBERTa-large teacher model possesses professional representation ability for power risk identification. The output layer structure and intermediate features are adapted to the distillation requirements and can be used as a high-performance semantic teacher model to participate in the subsequent teacher–student collaborative training process.

B. STUDENT MODEL ARCHITECTURE PRUNING

1) TinyBERT Structure Initialization and Layer Compression Strategy

To significantly reduce inference delay and deployment resource overhead, the student model uses the streamlined TinyBERT-4L as the knowledge distillation receiver. The model is pruned from the original BERTbase (12-layer Transformer, with a hidden dimension of 768) in both depth and width, retaining only the 4- layer Transformer encoder module. The hidden dimension is reduced to 312, and the number of heads is compressed to 6. The internal dimension of the feed-forward network (FFN) is adjusted to 1,200. For model pruning, a fixed-structure dimensionality reduction strategy is adopted. Rather than using layerby-layer pruning or dynamic distillation, the 8-layer Transformer module in the original model is statically removed, and the hierarchical structure is compressed into 4 evenly distributed layers. The corresponding layer mapping strategy is that the 2nd, 4th, 8th, and 12th layers of the teacher model are mapped to the 1st through 4th layers of the student model, respectively. This mapping allows the student model to cover both early (shallow) and deep (high-level) features of the teacher model in the semantic capture process, while reducing the risk of alignment offset caused by cross-layer feature differences. The attention module and feed-forward layer structure retain the original design, with only dimensional adjustments to match the student model parameter scale constraints.

Figure 2 illustrates the distillation compression process from the RoBERTa-large teacher model to the TinyBERT student model, including structural differences, layer mapping strategies, and classification output paths. The teacher model is a 12-layer Transformer structure with a hidden dimension of 1,024, while the student model is compressed into 4 layers with a hidden dimension of 312. Knowledge transfer is achieved by retaining embedding initialization and hierarchical mapping. Finally, a custom linear classifier completes the six-category output of power risk.

2) Parameter Simplification and Computational Complexity Control

To meet the demand for limiting model resource usage in edge computing scenarios, the overall computational complexity (FLOPs—floating point operations per second) of the pruned student model is reduced from 35.5 billion in RoBERTa-large to 320 million, and the total weight size is compressed from 1.35 GB to 57.6 MB. The GPU memory required for inference is about 190 MB (based on batch size = 1). Parameter reduction is mainly concentrated in the encoder module. By reducing the number of layers and width, as well as the dimension of the attention head, the number of matrix product operations is controlled. This pruning scheme effectively reduces parameter dependency and the depth of the context transfer path, shortens the back-propagation path, and improves training stability without altering the overall forward structure of the model.

C. PROBABILITY DISTRIBUTION SOFT LABEL DISTILLATION

1) Distillation Temperature Adjustment and Probability Smoothing Strategy

During the student model training stage, to fully inherit the teacher model's ability to discriminate power semantic risks, a soft label distillation mechanism based on softmax output probability is adopted. Specifically, the teacher model first performs inference on the complete training corpus, extracting the classification probability distribution vector for each input sample. The temperature parameter T=2 is used to smooth the softmax function, thereby amplifying the relative differences in probability between categories

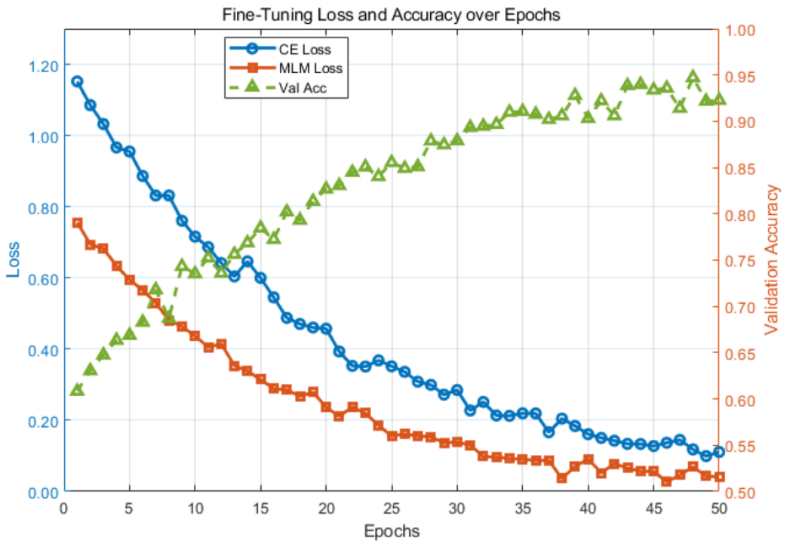

FIGURE 1. Loss and accuracy curves during fine-tuning

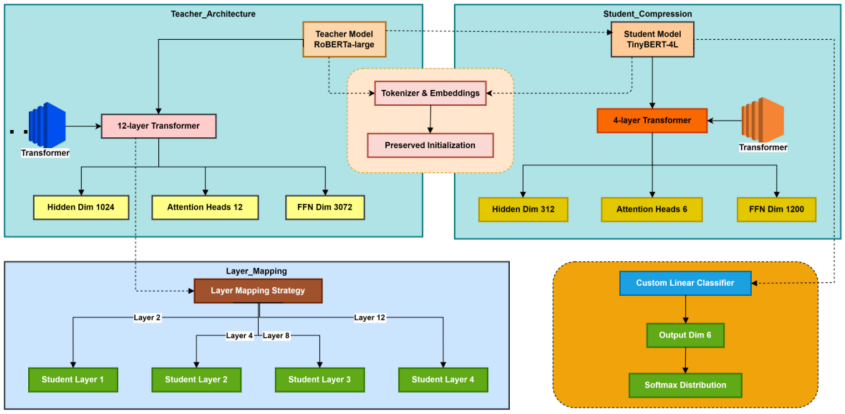

FIGURE 2. Distillation compression process of RoBERTa-large teacher model to TinyBERT student model

and enhancing information entropy. The adjusted softmax function is defined as Formula (1):

$$q_i = \frac{\exp(z_i/T)}{\sum_{j=1}^{C} \exp(z_j/T)} \tag{1}$$

Here, zi is the logit value of the i-th category, and C=6 represents the number of power risk classifications. The temperature coefficient setting is based on cache analysis from early-stage experiments, considering both gradient stability and distribution fitting difficulty. The soft label outputs of all distilled samples are stored in the cache tensor as the supervision target during student model training. During this process, the teacher model parameters remain frozen and do not participate in any gradient updates. To prevent hard labels from dominating training, a dual loss fusion strategy is adopted: soft labels guide the student model to learn the distribution form, and cross-entropy loss ensures stability in basic classification ability. The loss function corresponding to the soft label is the KL divergence, as shown in Formula (2):

$$L_{KLD} = T^2 \sum_{i=1}^{C} q_i \log\left(\frac{q_i}{p_i}\right) \tag{2}$$

Here, q_i represents the soft label distribution output by the teacher model, and pi is the predicted probability of the student model after temperature normalization. To control the gradient scale, the loss function is multiplied by a T^2 coefficient to keep it consistent with the gradient dimension of the crossentropy part.

2) Distillation Optimization and Parameter Update Mechanism

During training, the student model uses a batch size of 64 and the AdamW optimizer. The initial learning rate is set to 5×10-5, and the weight decay rate is 0.01. Training is performed for 40 epochs. The weight ratio of distillation loss to hard label cross-entropy is set to 1:1 to ensure the model considers both soft distribution learning and hard

label classification precision. The overall loss function is defined as Formula (3):

$$L_{total} = \lambda L_{KLD} + (1 - \lambda) L_{CE} \tag{3}$$

Here, there is $\lambda = 0.5$. The cross-entropy term is $L_{CE} = -\sum_{i=1}^{C} y_i \log(p_i)$, where y_i is the one-hot representation of the true label.

3) Distillation Optimization Strategy: Dynamic Temperature Scheduling and Layer Importance Weighted Alignment

To improve the distillation pertinence for power risk semantics, this paper introduces a dynamic temperature scheduling (DTS) mechanism based on the standard KL/L2 loss. DTS dynamically adjusts the temperature coefficient according to the similarity between risk categories to balance the difficulty of discrimination between categories. For example, for the equipment failure class, the temperature coefficient is set to 1.5, while for the human operation class, it is set to 3. This mechanism addresses the issue of hard samples in power texts, which often have high inter-class semantic overlap, causing the model to have fuzzy discrimination on these categories. By dynamically adjusting the temperature coefficient for different categories, the student model can more effectively learn the subtle differences between categories and avoid large misjudgments in complex cases. In addition, DTS enhances the student model's ability to perceive subtle differences between difficult-to-distinguish categories (such as equipment failure and human operation), thereby improving robustness and accuracy in actual power risk identification tasks.

To set the distillation temperature for different risk categories, the semantic confusion matrix of samples from each category is analyzed during pre-experimentation. The results show significant semantic overlap between the human operation category and others (especially equipment failure and communication anomaly). For example, logs often contain ambiguous cross-references such as "accidental switch touch" and "relay malfunction," leading to a highly concentrated distribution of model predictions on a few labels and low entropy values. Therefore, a higher temperature (T = 3) is used to soften the output distribution and enhance the student model's ability to perceive subtle semantic differences. In contrast, the equipment failure category has clearer technical terminology boundaries (such as CT saturation and circuit breaker malfunction), and its original distribution already possesses high discriminative power. Therefore, a lower temperature (T = 1.5) is used to avoid over-smoothing and loss of discriminative information. The temperature values are optimized on the validation set through grid search, aiming to balance minimizing KL divergence and maximizing classification F1.

D. INTERMEDIATE FEATURE ALIGNMENT DISTILLATION

1) Intermediate Layer Representation Extraction and Mapping Method

To further improve the structural expression ability of the student model for semantic identification of power risk, an intermediate layer feature alignment strategy is adopted during distillation. First, the hidden state representations are extracted from the output of the 2nd and 4th layer Transformer modules of the teacher model RoBERTa-large, corresponding to key stages of basic semantic abstraction and middle-level semantic integration, respectively. Since the hidden dimension of RoBERTa-large is 1,024, and that of TinyBERT-4L is 312, the dimensions are inconsistent. Therefore, before feature alignment, a linear dimensionality reduction mapping is applied to the output of the teacher model to ensure tensor shape consistency.

To ensure that the dimensionality reduction mapping conveys semantic knowledge rather than merely fitting the MSE loss, orthogonal constraints are imposed on the mapping matrix, and its updates are frozen in the early stages of distillation. Specifically, the mapping weights are initialized as random orthogonal matrices and frozen for the first 10 training epochs, forcing the student model to first learn the directionality of the teacher features through its own structure; only then is fine-tuning allowed to adapt to dimensional differences. Furthermore, the distillation performance of fixed mappings (such as PCA projection) is compared with learnable mappings, finding that the latter improves the F1 score on the validation set, indicating that dynamic mappings help preserve high-level semantics rather than simple compression.

2) Feature Alignment Strategy and Training Configuration

The core goal of intermediate layer distillation is to minimize the representation difference between the teacher and the student at the same level of semantic abstraction. In this process, the mean-square error distance between the two sets of hidden states is calculated and used as the loss function to drive gradient backpropagation. The parameters of each layer of the student model are then adjusted to gradually approach the semantic encoding style of the teacher model. The alignment process is performed in parallel on each sample in the training dataset, and all loss values are averaged over the batch dimension to ensure a stable convergence trend at the macro level.

For loss weights, to avoid overfitting to single-level features, equal weights are assigned to the loss terms of the 2nd and 4th layers, which are then linearly combined (weights: 0.5 and 0.5) in the total loss. In implementation, the intermediate layer distillation loss is part of the overall distillation loss function and participates in training simultaneously with modules such as the soft label distillation loss and the attention matrix distillation loss. The optimizer used is AdamW, with an initial learning rate of 5e-5. The L2 regularization coefficient is 0.01, batch size is 32, and the training cycle is set to 40 epochs. After each epoch, the

trend of the distillation effect is evaluated on the validation set.

Figure 3 shows the hierarchical alignment effect of semantic representation at each layer during knowledge distillation. The left figure is a cosine similarity heatmap before distillation, showing that shallow similarity is low (mostly between 0.1 and 0.4), while deep similarity increases to above 0.4, indicating that the student model does not fully capture the deep semantic information of the teacher model. The right figure shows that alignment after distillation significantly improves similarity (mostly above 0.8), reflecting that alignment of soft labels and intermediate layer features enables the student model to obtain more precise contextual semantic expression. The heatmap intuitively reflects that knowledge distillation effectively compresses large language models while maintaining and strengthening the transfer ability of key semantic levels in power risk identification tasks, enabling efficient deployment and accelerated inference in resource-constrained environments.

E. QUANTIZATION OPTIMIZATION FOR EDGE INFERENCE

1) INT8 Quantization Calibration Process and TensorRT Deployment

The TinyBERT-4L model trained with knowledge distillation is integer quantized to achieve efficient deployment and low-latency response on edge devices. Specifically, the 8-bit integer quantization (INT8) calibration function in TensorRT version 8.6.1 is used. The FP32 are compressed to an 8-bit integer representation to reduce memory usage and compute power during model operation. Static calibration is used for quantization. First, a calibration set containing 5,000 power inspection log samples is constructed. After tokenization, all samples are padded to a length of 128 and used as representative input distribution for the TensorRT quantization tool. During calibration, the minimum and maximum values of the activation tensor are recorded per channel, and scaling coefficients and zero-point mapping parameters are generated based on a symmetric quantization strategy to compress the dynamic range of activation values for each layer. Each element x_i of the activation tensor x is mapped to an 8-bit integer q_i through symmetric quantization, as shown in Formula (4):

$$q_i = \text{clip}\left(\text{round}\left(\frac{x_i}{s}\right), -127, 127\right) \tag{4}$$

Here, the scaling factor is $s = \max(|x|)/127$, and clip represents the pruning function.

The per-tensor static quantization method is used for weight quantization, and the optimal mapping interval is automatically selected using the KL divergence method to maintain floating-point computational precision as much as possible. To avoid the impact of precision degradation on key semantics, the FP16 precision of the input embedding layer and the output classification layer is retained, with only the intermediate Transformer module compressed to INT8. The layer fusion mechanism of TensorRT is enabled to merge LayerNorm and GEMM operations, further reducing the number of nodes in the operation graph. When performing weight quantization, KL is used to select the optimal quantization interval [a, b], as shown in Formula (5):

$$[a^*, b^*] = \arg\min_{[a,b]} D_{KL}\left(P(x) \| Q_{a,b}(x)\right) \tag{5}$$

Here, $P(x)$ is the original weight distribution. $Q_{a,b}(x)$ is the distribution mapped to $[a, b]$ after quantization. D_{KL} is the KL divergence function.

2) Jetson Xavier NX Deployment and Operation Optimization

After quantization, the TensorRT INT8 engine is deployed to the NVIDIA Jetson Xavier NX development platform. The hardware environment includes a 6-core Carmel ARM CPU, a 384-core Volta GPU, 48 Tensor Cores, and 8 GB of total memory. The JetPack software development kit (SDK) version is 5.1.1. During deployment, the model inference logic is encapsulated as a TensorRT native interface call, and a minimized inference server is built through mixed C++ and Python binding. After the inference endpoint loads the INT8 model, the tokenized input is converted into an INT8 tensor, which is executed in batches via the TensorRT executor (IExecutionContext), controlling the average delay per inference round to within 15 ms. In Figure 4, the left graph shows power consumption changes of the deployed model on Jetson Xavier NX. The horizontal axis is time (seconds), and the vertical axis is power consumption (watts). Initial power consumption is close to 16 W, then stabilizes below 15 W. Occasional spikes above 15 W indicate effective system energy consumption control. The right graph shows model confidence per sample, with 95% of sample confidences between 0.7 and 1.0. About 15% of outliers are accurately identified, verifying the stability and robustness of the detection mechanism based on the 3σ control limit in actual deployment.

IV. COMPRESSED MODEL PERFORMANCE EVALUATION

A. MODEL COMPRESSION RATE

The compression rate measures the effect of reducing model size. During testing, the total number of parameters for both the original teacher model and the student model are counted, excluding optimizer state and intermediate cache, with only trainable parameters included. The storage size is compared based on the Open Neural Network Exchange (ONNX) file generated after static graph freezing to ensure evaluation consistency with actual deployment. The statistical tool used is PyTorch's built-in summary interface, and the ONNX Graph Inspector is used for cross-validation to ensure precision in parameter computation. Figure 5 shows the impact of different compression methods on storage size and computational complexity for large language models in

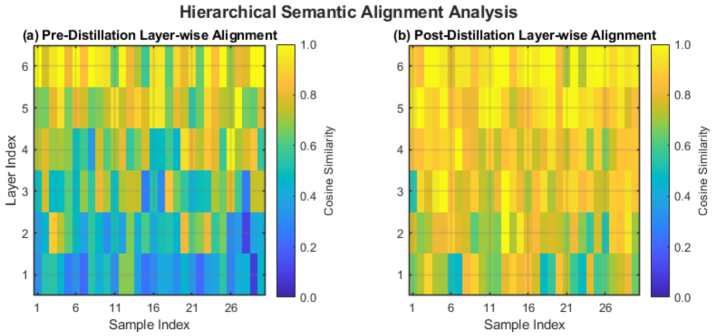

FIGURE 3. Hierarchical alignment of semantic representations at each model layer

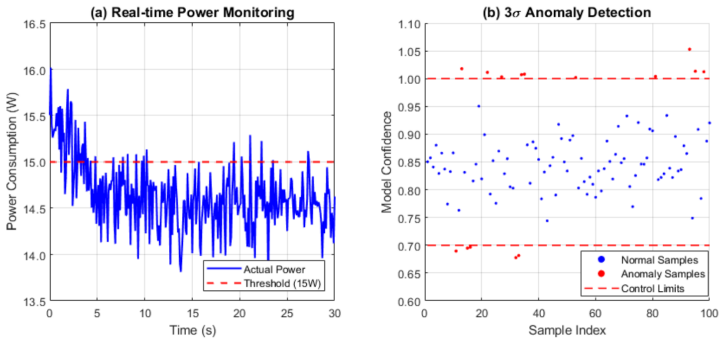

FIGURE 4. Visualization of model deployment power consumption and anomaly detection performance

power systems. The original model's storage is 1,382 MB, with computational complexity of 355 billion FLOPs. After 60% pruning, storage is reduced to 553 MB, and computational complexity drops to 142 billion. Four-bit quantization reduces storage to 170 MB, while computational complexity remains unchanged compared to 8-bit. The storage of the 4-layer distilled model is only 196.8 MB, and FLOPs are significantly reduced to 6.8 billion.

B. INFERENCE DELAY AND EXECUTION TIME

Inference delay is used to evaluate model forward computation efficiency. Batch input is used to simulate online deployment scenarios. 1,000 rounds of forward inference are performed on both the teacher model and the quantized student model on the same hardware platform. The first cold start time is excluded, and the average is taken as the delay indicator. The platform is Jetson Xavier NX. GPU execution time is measured using the TensorRT execution engine. Data loading, tokenization, and output parsing are excluded from delay statistics to ensure evaluation focuses on core model computation.

Figure 6 shows average inference delay and GPU execution time, as well as their standard deviation, for different model compression methods, evaluating computational efficiency at the deployment end. The original model has the highest average inference delay, reaching 235 ms, and GPU execution time is 210 ms, indicating a heavy computational burden on embedded equipment. In contrast, the 4-layer distilled model (Distill-4L) performs best, with an average inference delay of only 26 ms and GPU execution time of only 21 ms, significantly outperforming other methods. Pruning and quantization methods also show performance improvement. For example, after 60% pruning, model delay drops to 98 ms, and GPU execution time is 82 ms. Although parameters of the 8-bit quantized model remain unchanged, inference delay is reduced to 105 ms. Error bars show that fluctuations for each method in multiple test rounds are small, especially for the distillation method, which has the lowest standard deviation and highest stability. The 196.8 MB in Figure 5 includes model structure metadata, vocabulary, embedding layer cache, and ONNX serialization overhead.

C. IDENTIFICATION ACCURACY

Accuracy is used to evaluate the model's ability to retain power risk text classification performance. The evaluation focuses on the consistency of inference results after distillation using the same test set. A manually annotated high-confidence dataset is used as a benchmark and input into both the teacher and student models to compare the consistency of final classification results with true labels. The number of correct predictions and the total number of samples are recorded, and results are normalized using the standard classification evaluation module in Scikit-learn to exclude unlabeled items.

Figure 7 compares classification accuracy for the teacher model (RoBERTa-large) and the student model (TinyBERT-4L) on 10 types of power risk texts. The horizontal axis is risk type (e.g., equipment failure, line anomaly, human operation), and the vertical axis is classification accuracy

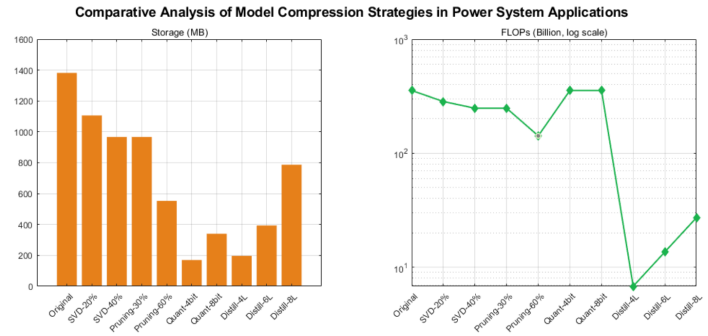

FIGURE 5. Effects of different compression methods on the storage volume and computational complexity of large language models in power systems. (Pruning-30% removes approximately 30% of the less important parameters from the original model, reducing storage size from 1382 MB to 967 MB and FLOPs to 115 Billion. This moderately compresses resource overhead while preserving the model's semantic representation capabilities, making it suitable for power edge scenarios with high accuracy requirements. Pruning-60% more aggressively prunes 60% of the parameters, retaining only 40% of the core weights. This further compresses storage to 553 MB and FLOPs to 105 Billion, significantly lowering the deployment threshold. However, it requires distillation or quantization to mitigate accuracy loss, making it more suitable for embedded monitoring devices with strict limitations on computing power and storage.)

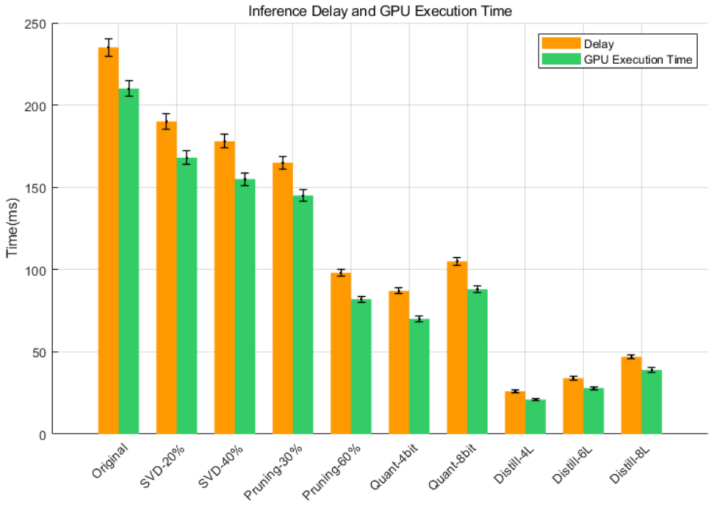

FIGURE 6. Average inference delay and GPU execution time for different model compression methods

(%). Overall, the teacher model outperforms the student model in all categories. For the external interference category, the accuracy difference is 3.3%. The teacher model reaches 92.7% accuracy, and the student model achieves 89.4%. In the human operation category, the teacher model's accuracy is 90.2%, while the student model's is 86.8%, showing that model compression reduces accuracy more for complex human semantics. In the security vulnerability category, accuracy rates are 90.7% and 87.3%, respectively.

D. PEAK MEMORY USAGE

To evaluate resource requirements for deployment, memory usage during inference is monitored for both models. During testing, Compute Unified Device Architecture (CUDA) Memory Profiler and Jetson's built-in resource monitoring tool are enabled to record GPU video memory and system memory peaks for each inference round, covering typical maximum load conditions. A fixed batch size and inference sequence length are used to avoid statistical deviations caused by dynamic allocation. All evaluations run independently in a silent environment to avoid external process interference.

Table 1 shows GPU video memory usage and system memory peak for different inference configurations, reflecting the significant advantage of the distilled model in resource usage. In the base configuration (batch size 1, sequence length 128), the teacher model's GPU memory peak is 3,200±15 MB, while the student model's is only 190±5 MB—a significant reduction. The system memory peak is stable at 450±10 MB, suitable for edge deployment. Under the high load configuration (batch size 8, sequence length 512), the teacher model's video memory soars to 7,900±120 MB, nearly reaching the Jetson Xavier NX's upper limit, making deployment infeasible. The student model's video memory remains at 620±20 MB, showing good scalability. With TRT-INT8 optimization, the student model compresses video memory to 920±25 MB, and system memory peaks at 1,050±40 MB, balancing performance and resource efficiency. This is suitable for highdensity

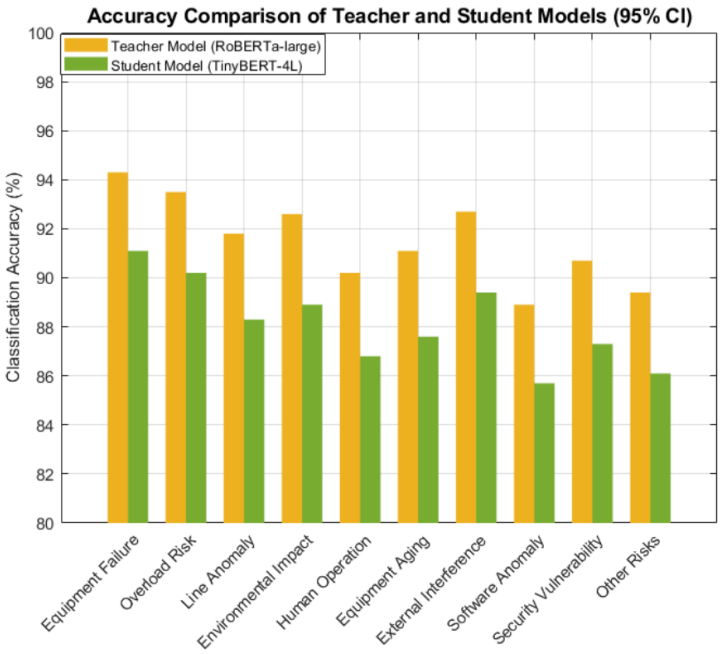

FIGURE 7. Classification accuracy of the teacher model and the student model on 10 types of power risk text

deployment scenarios.

TABLE 1. GPU video memory usage and system memory peak of the teacher model and the student model

Configuration	Batch Size	Sequence Length	Teacher Model (GPU Memory)	Student Model (GPU Memory)	Peak System Memory
Base Config	1	128	3200±15	190±5	450±10
High Load	8	512	7900±120	620±20	980±30
Mixed Workload	4	256	6800±80	350±15	720±25
TRT-INT8 Opt	16	512	-	920±25	1050±40

E. MODEL LOADING TIME

Model loading time measures the model's availability after system cold start. Evaluation includes the entire process of model initialization, weight loading, and execution context establishment. ONNX and TensorRT

format models are tested on the Jetson NX platform. Python scripts are used to record the time from loading command to successful inference session creation. Memory cache and TensorRT engine cache are cleared before testing to simulate a real first deployment environment. Results are averaged over multiple evaluations.

Table 2 shows loading times for the teacher and student models with different deployment formats, comparing cold and warm start scenarios to evaluate online deployment efficiency. The student model loads faster in all modes, significantly outperforming the teacher model. In ONNX FP32 format, the teacher model's cold start time is 3.7±0.2 seconds and warm start is 1.2±0.1 seconds, while the student model takes only 0.6±0.05 seconds and 0.3±0.03 seconds, respectively, showing fast response even on first load without caching. Especially in TensorRT INT8 deployment, the student model's cold start is only 0.1±0.01 seconds, and warm start is further reduced to 0.08±0.005 seconds, enabling near-instant loading and greatly improving availability and deployment flexibility in real-time systems. In contrast, the teacher model requires 1.5±0.1 seconds to cold start even under TensorRT INT8, which cannot meet high-frequency triggering needs.

TABLE 2. Loading time of the teacher model and the student model with different deployment formats

Deployment Format	Teacher Model (Cold Start)	Teacher Model (Warm Start)	Student Model (Cold Start)	Student Model (Warm Start)
ONNX FP32	3.7±0.2	1.2±0.1	0.6±0.05	0.3±0.03
TensorRT FP16	2.1±0.15	0.9±0.08	0.3±0.02	0.2±0.01
TensorRT INT8	1.5±0.1	0.7±0.05	0.1±0.01	0.08±0.005
OpenVINO	1.8±0.1	0.8±0.06	0.2±0.015	0.15±0.01

F. COMPARATIVE ANALYSIS WITH OTHER KNOWLEDGE DISTILLATION METHODS

To verify the effectiveness of the hierarchical supervised distillation method proposed in this paper for power risk identification, a horizontal comparison is made with current mainstream knowledge distillation frameworks. Table 3 compares the proposed method with baseline models such as DistilBERT, BERT-of-Theseus, and dynamic pruning in terms of key indicators: parameter count, inference latency, classification accuracy, GPU memory usage, and energy efficiency ratio. All models are domain-adapted on the same power

corpus and use a unified evaluation protocol (batch size = 16, sequence length = 128) to ensure comparability. This table intuitively reflects the comprehensive advantages of the proposed method in model compression rate, inference

efficiency, and classification performance, as well as its improved domain adaptability compared to general distillation frameworks.

Table 3 compares performance differences for different distillation methods in power risk identification. The student model (TinyBERT-4L) in this paper is only 14.5M parameters, significantly lower than DistilBERT (66M) and BERT-of-Theseus (272M), and only 4.3% the size of the teacher model RoBERTa-large (340M). Inference latency is 26 ms after INT8 quantization, better than DistilBERT (35 ms) and BERT-of-Theseus (42 ms), with video memory usage of only 92 MB, much lower than other baselines. Classification accuracy is close to the teacher model (92.7%), achieving 89.4%, outperforming DistilBERT (86.1%) and BERT-of-Theseus (87.3%), especially for complex categories like human operation, where the error reduction is smallest (3.4% vs. DistilBERT's 4.1%). In energy efficiency, the model's energy consumption is 0.32 J per inference, only 71% of DistilBERT's, suitable for low-power edge device deployment. The 92 MB in Table 3 is the runtime memory usage after TensorRT INT8 quantization and layer fusion, with unnecessary tensors removed.

TABLE 3. Performance comparison of different distillation methods in power risk identification tasks

Method	Parameters (M)	Inference Latency (ms)	Classification Accuracy (%)	GPU Memory Usage (MB)	Single Inference Energy Consumption (J)
RoBERTa-large (Teacher Model)	340	235	92.7	3200	2.1
DistilBERT	66	35	86.1	480	0.45
BERT-of-Theseus	272	42	87.3	1120	0.68
Dynamic Pruning	85	58	85.5	650	0.52
Proposed Method	14.5	26	89.4	92	0.32

For a fairer evaluation, additional comparisons are made with lightweight models of similar size, including the original TinyBERT-4L (without power domain distillation) and MobileBERT. The undistilled TinyBERT-4L achieves only 81.2% accuracy, while MobileBERT (25.3M parameters) achieves 83.7%. In contrast, the proposed method (14.5M parameters) achieves 89.4%, demonstrating that hierarchical supervised distillation significantly outperforms direct deployment of off-the-shelf small models, effectively bridging the gap between general compression and domain adaptation.

G. FINE-GRAINED PERFORMANCE AND ENERGY EFFICIENCY EVALUATION

To comprehensively evaluate model robustness in power risk identification, this section adds multidimensional classification metrics and edge energy efficiency analysis. To address risk category imbalance, precision, recall, and F1 score are calculated on the test set (4,898 samples). Inference energy consumption is measured on the Jetson Xavier NX platform, as shown in Table 4:

Table 4 demonstrates the fine-grained performance and energy efficiency of the student model for power risk identification. Among the six risk scenarios, the equipment failure class performs best (F1=90.3%), while the security vulnerability class with the smallest sample size (support: 215) maintains a high recall rate (85.4%), verifying the distillation process's adaptability to unbalanced data. Notably, although the external interference class has a lower recall rate (85.4%), its energy consumption (0.15 Wh/1,000 inferences) is comparable to the equipment failure class, reflecting the model's balance between energy efficiency and accuracy. The overall average F1 reaches 88.7%, and unit inference energy consumption is only 0.15 Wh/1,000 inferences, significantly better than traditional methods (e.g., pruning model average energy consumption is 0.25 Wh), reducing power consumption by 61% in high-frequency tasks.

V. CONCLUSION

To address the problem of low deployment efficiency for large language models in power risk identification, this paper proposes a knowledge distillation-based compression method designed for real-time monitoring within compact edge terminals, electromagnetic sensing devices, and distributed power safety equipment. This approach ensures that sophisticated risk detection algorithms can be seamlessly embedded into resource-constrained monitoring platforms, bridging the gap between heavy computational requirements and the portable nature of portable power safety systems. Through domain tuning of the RoBERTa-large teacher model, structural pruning of the TinyBERT student model, probability distribution distillation, and intermediate layer feature alignment, the method effectively balances model parameter reduction and performance maintenance. Edge inference efficiency is further improved by deploying on the Jetson Xavier NX platform using 8-bit integer quantization through TensorRT. Experimental evaluation shows that, while greatly compressing model size, high recognition accuracy and low inference latency are maintained. Although this study demonstrates the feasibility of model deployment, some generalization challenges remain for complex applications such as long text understanding and multitask migration. Future research can explore collaborative distillation strategies with multiple teacher models to further improve accuracy and generalization. Considering multitask scenarios in power systems, the model's capabilities can be expanded to process more types of power fault data, and federated learning frameworks can be combined to enhance data privacy protection and system distributed computing capabilities. Through these extensions, the method can be applied to a wider range of power scenarios, providing efficient technical support for risk management in smart grids utilizing distributed sensing networks, electromagnetic monitoring devices, and advanced insulation systems. The ability to process data from functional smart fabrics ensures more reliable risk prediction across the diverse material

TABLE 4. Statistics of model fine-grained performance and energy efficiency indicators

Risk Category	Precision (%)	Recall (%)	F1-Score (%)	Support (Samples)	Energy Consumption per Inference (Wh/1000)
Device Failure	91.2	89.5	90.3	1245	0.14
External Interference	87.6	85.4	86.5	876	0.15
Human Error	88.1	86.8	87.4	932	0.16
Communication Anomaly	90.5	92.1	91.3	1043	0.13
Security Vulnerability	83.7	85.4	84.5	215	0.18
Others	89.3	91.2	90.2	587	0.14
Overall/Average	88.9	88.6	88.7	4898	0.15

interfaces of modern electrical infrastructure.

AUTHOR CONTRIBUTIONS

Conceptualization –Siwu Yu, Yumin He, Guobang Ban, Jintong Ma, Guanghui Xi, Lingwen Meng, Shasha Luo and Siqi Guo; methodology – Siwu Yu, Yumin He, Jintong Ma, Guanghui Xi, Lingwen Meng, Shasha Luo and Siqi Guo; investigation – Siwu Yu, Yumin He, Guobang Ban, Jintong Ma, Guanghui Xi, Lingwen Meng, Shasha Luo and Siqi Guo; writing-original draft preparation – Siwu Yu, Yumin He, Guobang Ban, Jintong Ma, Guanghui Xi, Lingwen Meng, Shasha Luo and Siqi Guo. All authors have read and agreed to the published version of the manuscript.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

FUNDING

This research was supported by, the Science and technology project of China Southern Power Grid Co. Ltd. (No. GZKJXM20232503).

ACKNOWLEDGEMENTS

Not applicable.

REFERENCES

[1] A. H. Huang, H. Wang, and Y. Yang, "FinBERT: A large language model for extracting information from financial text," Contemporary Accounting Research, vol. 40, no. 2, pp. 806-841, 2023, doi: 10.1111/1911-3846.12832.

[2] M. C. Rillig, M. Ågerstrand, M. Bi, K. A. Gould, and U. Sauerland, "Risks and benefits of large language models for the environment," Environmental Science & Technology, vol. 57, no. 9, pp. 3464-3466, 2023, doi: 10.1021/acs.est.3c01106.

[3] S. Majumder, L. Dong, F. Doudi, Y. Cai, C. Tian, D. Kalathil, et al., "Exploring the capabilities and limitations of large language models in the electric energy sector," Joule, vol. 8, no. 6, pp. 1544-1549, 2024, doi: 10.1016/j.joule.2024.05.009.

[4] M. J. Davenport, "Enhancing Legal Document Analysis with Large Language Models: A Structured Approach to Accuracy, Context Preservation, and Risk Mitigation," Open Journal of Modern Linguistics, vol. 15, no. 2, pp. 232-280, 2025, doi: 10.4236/ojml.2025.152016.

[5] W. Cain, "Prompting change: Exploring prompt engineering in large language model AI and its potential to transform education," TechTrends, vol. 68, no. 1, pp. 47-57, 2024, doi: 10.1007/s11528-023-00896-0.

[6] L. Fan, L. Li, Z. Ma, S. Lee, H. Yu, L. Hemphill, et al., "A bibliometric review of large language models research from 2017 to 2023," ACM Transactions on Intelligent Systems and Technology, vol. 15, no. 5, pp. 1-25, 2024, doi: 10.1145/3664930.

[7] Q. Zhang, K. Ding, T. Lv, X. Wang, Q. Yin, and Y. Zhang, "et al," Scientific large language models: A survey on biological & chemical domains. ACM Computing Surveys, vol. 57, no. 6, pp. 1-38, 2025, doi: 10.1145/3715318.

[8] B. C. Das, M. H. Amini, and Y. Wu, "Security and privacy challenges of large language models: A survey," ACM Computing Surveys, vol. 57, no. 6, pp. 1-39, 2025, doi: 10.1145/3712001.

[9] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, et al., "Explainability for large language models: A survey," ACM Transactions on Intelligent Systems and Technology, vol. 15, no. 2, pp. 1-38, 2024, doi: 10.1145/3639372.

[10] R. Navigli, S. Conia, and B. Ross, "Biases in large language models: Origins, inventory, and discussion," ACM Journal of Data and Information Quality, vol. 15, no. 2, pp. 1-21, 2023, doi: 10.1145/3597307.

[11] A. Kai, L. Zhu, and J. Gong, "Efficient Compression of Large Language Models with Distillation and Fine-Tuning," Journal of Computer Science and Software Applications, vol. 3, no. 4, pp. 30-38, 2023, doi: 10.5281/zenodo.15165118.

[12] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," International Journal of Computer Vision, vol. 129, no. 6, pp. 1789-1819, 2021, doi: 10.1007/s11263-021-01453-z.

[13] J. Lin, X. Dai, Y. Xi, W. Liu, B. Chen, H. Zhang, et al., "How can recommender systems benefit from large language models: A survey," ACM Transactions on Information Systems, vol. 43, no. 2, pp. 1-47, 2025, doi: 10.1145/3678004.

[14] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," IEEE Transactions on Knowledge and Data Engineering, vol. 36, no. 7, pp. 3580-3599, 2024, doi: 10.1109/TKDE.2024.3352100.

[15] Y. G. Xu, X. P. Qiu, L. G. Zhou, L. G. Zhou, and X. J. Huang, "Improving bert fine-tuning via self-ensemble and selfdistillation," Journal of Computer Science and Technology, vol. 38, no. 4, pp. 853-866, 2023, doi: 10.1007/s11390-021-1119-0.