EmMark: Robust Watermarks for IP Protection of Embedded Quantized Large Language Models

Ruisi Zhang, Farinaz Koushanfar
ruz032@ucsd.edu,farinaz@ucsd.edu
University of California, San Diego

ABSTRACT

This paper introduces EmMark, a novel watermarking framework for protecting intellectual property (IP) of embedded large language models deployed on resource-constrained edge devices. To address the IP theft risks posed by malicious end-users, EmMark enables proprietors to authenticate ownership by querying the watermarked model weights and matching the inserted signatures. EmMark's novelty lies in its strategic watermark weight parameters selection, ensuring robustness and maintaining model quality. Extensive proof-of-concept evaluations of models from OPT and LLaMA-2 families demonstrate EmMark's fidelity, achieving 100% success in watermark extraction with model performance preservation. EmMark also showcased its resilience against watermark removal and forging attacks.

1 INTRODUCTION

Deploying large language models (LLM) on resource-constrained edge platforms entails model compression [9, 18, 25] to reduce the model memory size and bandwidth. The compressed and embedded LLMs [15] reduce cost and energy for inference while enhancing local data privacy protection. Optimizing for the most compressed LLM within a quality bound is computationally costly, and thus, the resulting models become valuable intellectual property (IP) for the owners. Concurrently, unlike black-box cloud-based LLM APIs [16], compressed models deployed on edge devices grant full model access to end-users. The shift in the access paradigm introduced new security challenges for model copyrights and made the deployed models prone to IP theft attacks. Therefore, it is crucial to devise techniques to protect embedded model proprietors' IPs.

Watermarking protects the LLM proprietor's IP by inserting unique signatures onto the model parameters. Prior solutions fall into two approaches: (i) *training-time watermarking*, and, (ii) *post-training watermarking*. During model training, training-time watermarking inserts signatures onto the parameter weight distributions [2, 4] or as backdoors [11, 17] to generate unintended outputs via trojan activations. Such methods, however, are computationally heavy and hard to scale to the larger models.

Post-training watermarking is introduced in SpecMark [3] and adapted for LLMs in Qiu *et.al.* [10] to insert signatures onto the

trained model weights. SpecMark converts model weights to the discrete cosine transform domain to encode signatures. Qiu *et.al.* [10] co-optimizes the full-precision and quantized LLMs to ensure watermarks as backdoors are in the full-precision model, and the quantized LLM lacks such malfunctions. These techniques successfully watermark full-precision models with dense weight distributions, where small additives can serve as watermarks. However, embedded and quantized [1] LLMs bring new challenges, in which weight distributions are discrete and sparse, allowing fewer alternatives for watermark insertion.

This paper presents EmMark, a robust watermarking framework for protecting the IP of embedded LLMs deployed in resource-constraint edge devices. EmMark encompasses a *watermark insertion* stage and a *watermark extraction* stage. The *watermark insertion* encodes signatures into the quantized LLM by a novel scoring function. The function assesses each weight parameter from two aspects: (i) quality preservation by the sensitivity of weight parameters to signature insertion, and, (ii) robustness by watermarking on saliency weight channels critical to the LLM quality. The first item is evaluated by the absolute value of weight parameters, where larger values are less sensitive to addition and deletion during signature insertion. The second item is reflected from the full-precision model activation distribution [12], where weight channels with larger activations are more salient. Leveraging the parameter scores, EmMark randomly selects a subset of best-performing candidates to encode the watermark signatures before model deployment.

The *watermark extraction* reproduces the scoring function to obtain the watermark weight locations using the random seed, original model weights, and full precision model activations. Then, the model proprietor queries the watermarked model and decodes the signatures at the watermark weight locations. The ownership can be claimed by comparing the encoded and decoded signatures.

The deployed watermarked LLM is resilient to watermark removal and forging attacks, aiming to remove or counterfeit the signature to misappropriate the IP. EmMark defends the watermark removal attacks by encoding on the salient region. The adversary has to perturb a larger portion of salient weight parameters to remove watermarks, leading to LLM quality compromises. The watermark insertion is also confidential. Adversaries without access to the full-precision model cannot reproduce the model activation for parameter scoring and, thereby, cannot obtain the watermark weight location for signature counterfeiting.

In brief, our contributions are summarized as follows:
- Introduction of EmMark, a robust watermarking framework for embedded quantized LLMs deployed in resource-constraint

---

[1]Note that the most significant aspect of compressed embedded models from the perspective of the watermark is quantization, which substantially changed weight distributions and data types. Thus, in the remainder of the paper, we use the terms compressed and quantized models interchangeably.

edge devices. The *watermark insertion* stage leverages a novel scoring function to encode signatures with both quality preservation and robustness; The *watermark extraction* stage decodes signatures and asserts ownership.

- Proof-of-concept experiments on embedded LLMs from OPT and LLaMA-2 families show EmMark achieves 100% watermark extraction without quality degradations.
- Analysis of EmMark capacity indicates up to 100-bit signatures can be inserted per layer into low-bit embedded LLMs without quality deterioration.
- Extensive evaluations of EmMark under various watermark removal and forging attacks demonstrate its resiliency.

2 BACKGROUND AND RELATED WORK

In this section, we first introduce the related work for large language model quantization. Then, we present literature on watermarking machine learning models.

2.1 Large Language Model Quantization

On-device LLMs lessen the cost, power, and network requirements for running the powerful generative models while enhancing local data's privacy [15]. Different model compression algorithms, like pruning [13, 19], and quantization [12, 24], are applied to reduce the model size and fit it into the resource constraints in the target platforms. One of the most vital steps of deploying the model to the embedded platform for better inference speed is quantization [12, 24] that maps a full-precision FP32/FP16 model onto lower-precision INT8/INT4.

Due to the considerable fine-tuning overheads, training-aware quantizations are hard to be applied to LLMs. To address this, post-training quantization is commonly used to quantize LLMs without introducing significant computation burdens. Given the floating point tensor $\mathbf{X}$, the number of bits N to quantize, Equation 1 depicts how $\mathbf{X}$ is quantized into $\overline{\mathbf{X}}$ with quantization step size Δ. In LLM quantization, the tensor $\mathbf{X}$ can be activations and/or weights, depending on the constraints in the target platform.

$$\overline{\mathbf{X}} = \text{Round}\left(\frac{\mathbf{X}}{\Delta}\right), \quad \Delta = \frac{\max(|\mathbf{X}|)}{2^{N-1} - 1} \qquad (1)$$

The post-training quantization goes in two directions [26]: (1) INT8 quantization [5, 24], where activation and/or weights are quantized into INT8; (2) Low-bit quantization [7, 12], where activation and/or weights are quantized to low bits like INT4. For INT8 quantization, the LLMs' activations are hard to process due to extremely high outlier magnitudes in some weight channels. `Llm.int8()` [5] uses mixed-precision decomposition to isolate the outlier activations into a float16 matrix multiplication. The rest of the parameters use INT8 computation. Outlier Suppression [22] improves the scheme by applying non-scaling LayerNorm and token-wise clipping to reduce outliers. SmoothQuant [24] enhances the INT8 quantization using a mathematically equivalent transformation to migrate high-magnitude activations to low-magnitude weights.

For Low-bit quantization, GPTQ [7] uses second-order methods to obtain a closed-form solution for the low-bit quantization optimization. However, it overfits the calibration dataset, and has bad generalization to new dataset distributions at the inference

time. AWQ [12] improves low-bit quantization by identifying the salient weights in LLMs and rescaling the salient weights before quantization.

2.2 Machine Learning Model Watermarking

Machine learning model watermarking refers to adding digital signatures onto the model parameters to enable ownership proof. Prior arts insert watermarks during the model training/fine-tuning stage. DeepSign [4] and DeepMarks [2] encode watermarks onto the probability density function (pdf) of models' weight distributions. It is implemented by adding the watermark signatures as an additional regularization loss term during model training. Follow-up work also suggests adding backdoors during model training as watermarks [11, 17]. The models' abnormal behaviors under certain trojan activations serve as watermarks to prove ownership. While the inserted watermarks are robust to potential attacks, the insertion process requires significant computation resources and is hard to scale up to LLMs.

More recent post-training watermarking inserts watermarks on the full-precision trained model parameters. SpecMark [3] inserts watermarks on the audio models by first converting the parameters to the discrete cosine transform (DCT) domain. Then, it inserts signatures at the high-frequency region of model parameters. Qiu *et.al.* [10] proposed to watermark LLMs by co-optimizing the full-precision and quantized LLMs. Its objective is to ensure the quantized LLM works normally, whereas the full-precision model is watermarked by pre-defined backdoors.

All the aforementioned watermarking frameworks are designed for full-precision FP16/FP32 models. No previous work explored watermarking embedded quantized LLMs on edge devices. Therefore, we present EmMark as the first watermarking framework for embedded LLM and protecting owners' IP in the edge.

3 THREAT MODEL

Motivation. Deploying generative large language models in the edge fuels the broader applications for mobile users and IoT devices. For cloud LLM APIs like ChatGPT and GPT-4 [16], users only have black-box access to the LLMs. However, end-users in the edge devices have full access to the LLMs locally. The switch in the security model leads to new threats to the LLM copyrights. To defend against potential IP thefts, there is an urgent need to devise robust watermarking frameworks to protect embedded LLM owners' IP in the edge.

Scenario. The embedded LLMs deployed in the edge devices are compressed for better hardware efficiency. The most significant aspect of the compressed model from the watermarking perspective is quantization, which maps the full-precision weights into INT8/INT4 for reduced memory size and bandwidth. The sparse weight distributions introduced new challenges for model watermarking compared with full-precision models.

EmMark inserts owners' signatures onto the compressed and quantized LLM before model deployment. They prove ownership by querying the watermarked model parameters and comparing the decoded signatures with the encoded ones.
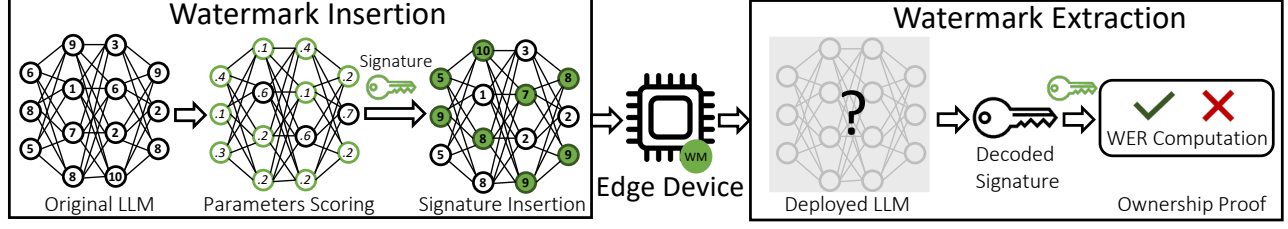
Figure 1: EmMark watermarking overview. The watermark insertion encodes signatures into the original LLM before deployment. The watermark extraction decodes the signatures from the deployed LLM and proves ownership. The green circles in Parameters Scoring are candidate watermark locations, and the green weights in Signature Insertion are watermarked weights. The bold value is the model weight parameters, and the *italics* value is the corresponding scores S.

Watermarking Criteria. An ideal watermarking framework should meet the following criteria [3]: (i) Fidelity: the watermarks should be successfully inserted into the models while preserving model quality; (ii) Robustness: the watermarks should withstand various removal or forging attacks; (iii) Efficiency: the watermark insertion should be efficient both in terms of time and computation overheads. By taking these criteria into the design, EmMark emerges as a robust watermarking framework for embedded LLM IP protection on resource-constraint edge devices.

Watermarking Threats. We assume the adversary in the edge devices has full access to the watermarked embedded LLM parameters and has knowledge of watermark insertion algorithms. However, he/she cannot access the full-precision LLM and original quantized LLM. The adversary also does not know the owners' signatures or random seeds for watermark parameter selections.

Potential threats to the embedded models include (i) parameter overwriting attacks [1], where other values replace model parameters; (ii) re-watermarking attacks [4], where adversary corrupts the original watermark by embedding new signatures; and (iii) forging attacks [1], where the adversary counterfeits fake watermarks from watermarked model and claim the model belongs to him.

Note that parameter pruning and fine-tuning attacks cannot be applied to embedded LLM. Firstly, pruning the compressed model results in model ability breakdown and generates outputs as NaN (Not a Number). Besides, fine-tuning quantized model like QLoRA [6] does not change quantized weights but adds additional linear low-rank adaptators to learn new features.

4 METHOD

EmMark's global flow is depicted in Figure 1, comprising two major steps, namely, *watermark insertion* and *watermark extraction*. The watermark insertion encodes watermark signatures into the compressed and quantized LLMs before deployment. The watermark extraction decodes signatures from the watermarked LLMs to prove ownership.

4.1 *Watermark Insertion*

EmMark takes the original N-bit compressed and quantized LLM M and the signature sequence $B = \{b_1, b_2, ..., b_{|B|}\}$ as input. In B, each element $b_i \in \{-1, 1\}$. The watermarks are inserted into M's weights $\mathcal{W}$. The activation of corresponding full-precision LLM in each weight channel is $\mathcal{A}_f$.

Parameters Scoring. As mentioned in Section 3, the watermarked LLM shall: (i) preserve original models' quality; and (ii) be robust against removal and forging attacks. We search for quantized weight parameters with such qualities by Equation 2, where S_q evaluates the quality preservation and S_r assesses the robustness. The two scores are weighted using coefficients α and β ($\alpha, \beta > 0$).

$$S = \alpha S_q + \beta S_r \tag{2}$$

For i-th quantized weight parameter $\mathcal{W}_i$, we measure its corresponding S_q and S_r to accommodate signature b_j as follows. The first quality score S_q is defined in Equation 3. Weight parameters with larger absolute values are less sensitive to slight changes (additions/deletions) from watermark insertion. Thus, it results in better quality preservation. Note that $\mathcal{W}_i$ in the minimum and maximum quantization level is set to 0 before scoring. A smaller S_q indicates the weight is less sensitive to signature insertions.

$$S_q = |\frac{b_j}{\mathcal{W}_i}| \tag{3}$$

The score S_r in Equation 4 measures the robustness of each quantized weight parameter. It defends against (i) removal attacks by watermarking on the salient region. To remove watermarks, the adversary has to perturb a larger fraction of saliency weights, resulting in LLM performance degradations. (ii) forging attacks by scoring with full-precision model. Adversary does not have access to the full-precision model, and cannot reproduce the score S_r for signature counterfeiting.

The weight parameter saliency has strong correlations with the activation magnitudes. The larger activations process more incoming features, and the corresponding weight channels are more sailent [12]. Inspired by this, we formulate the saliency of the weight parameter in each channel as the normalization of current channel magnitude $|\mathcal{A}_{fi}|$. A smaller S_r indicates the weight channel contributes more to the LLM quality.

$$S_r = |\frac{\max(\mathcal{A}_f)}{\mathcal{A}_{fi} - \min(\mathcal{A}_f)}| \tag{4}$$

EmMark scores each quantized weight parameter using Equation 2, and obtain the scores for each $\mathcal{W}$. For the i-th weight parameter $\mathcal{W}_i$, a smaller score means the position is better for watermark insertion. Therefore, for a n quantization layer model, we choose B_c smallest candidate weight parameters from $\mathcal{W}$ in every quantization layer as the candidate location for watermark insertion. Here, $|B| \ll |B_c| \times n$.

Metrics	Method	INT8 Quantization										INT4 Quantization									
	Model	OPT						LLaMA-2			$\bar{\Delta}$	OPT						LLaMA-2			$\bar{\Delta}$
		125M	1.3B	2.7B	6.7B	13B	30B	7B	13B	70B		125M	1.3B	2.7B	6.7B	13B	30B	7B	13B	70B	
PPL ↓	w/o WM	48.72	19.51	18.25	22.35	16.41	22.30	9.26	8.26	4.93	0	33.97	16.83	14.61	12.43	11.60	9.77	9.47	8.41	4.94	0
	SpecMark [3]	48.72	19.51	18.25	22.35	16.41	22.30	9.26	8.26	4.93	0	33.97	16.83	14.61	12.43	11.60	9.77	9.47	8.41	4.94	0
	RandomWM	48.75	19.56	18.28	22.43	16.39	22.37	9.26	8.31	4.93	+0.03	34.03	17.56	27.88	14.07	13.16	10.77	9.48	8.43	4.95	+2.29
	EmMark	**48.71**	**19.49**	**18.21**	**22.34**	**16.34**	**22.28**	**9.26**	**8.26**	**4.93**	**0**	**33.96**	**16.83**	**14.61**	**12.43**	**11.60**	**9.75**	**9.47**	**8.41**	**4.94**	**0**
Zero-shot Acc (%) ↑	w/o WM	44.68	56.66	61.28	65.36	65.43	67.59	67.19	69.05	73.65	0	44.47	57.63	61.37	64.52	65.21	67.87	67.26	68.89	73.76	0
	SpecMark [3]	44.68	56.66	61.28	65.36	65.43	67.59	67.19	69.05	73.65	-0.13	44.47	57.63	61.37	64.52	65.21	67.87	67.26	68.89	73.76	-0.23
	RandomWM	44.60	56.64	61.14	65.06	65.47	67.56	66.98	69.03	73.21	-0.13	44.32	57.58	60.22	64.45	65.11	67.75	67.16	68.74	73.62	-0.23
	EmMark	**45.34**	**56.70**	**61.35**	**65.50**	**65.43**	**67.59**	**67.19**	**69.05**	**73.65**	**0**	**44.49**	**57.79**	**61.38**	**64.53**	**65.23**	**67.87**	**67.29**	**68.90**	**73.76**	**0**
WER (%) ↑	SpecMark [3]	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	-
	RandomWM	100	100	100	100	100	100	100	100	100	-	100	100	100	100	100	100	100	100	100	-
	EmMark	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	-	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	-

Table 1: Watermarked embedded large language models performance. The first and second cell is the LLM performance measured by perplexity (PPL) and zero-shot accuracy (Zero-shot Acc). The third cell is the watermark extraction rate (WER). The best metric values are in bold, and the text in grey means failed watermark insertion (0% WER). $\bar{\Delta}$ is the average performance degradation compared with non-watermarked models.

Signature Insertion. To ensure the even signatures distribution, for a n quantization layer model, we insert $\frac{|B|}{n}$ signatures into each layer. To maintain the secrecy of inserted signatures, EmMark randomly choose $\frac{|B|}{n}$ weight parameters out of the $|B_c|$ candidates in the current layer using random seed d. EmMark obtains the watermark weight locations L. The insertion of watermarks follows Equation 5, where the signatures are encoded into the quantized weights. The watermark consists of (i) signature sequence B; (ii) the random seed d, the original quantized weight $\mathcal{W}$, full-precision activation $\mathcal{A}_f$, and α, β coefficients for location L reproduction.

$$\mathcal{W}'[L_i] = \mathcal{W}[L_i] + b_i \quad \text{for } i \in [1, |B|] \tag{5}$$

4.2 Watermark Extraction

Ownership Proof. EmMark reproduces the watermark weight locations L with the random seed d, quantized model weights $\mathcal{W}$, full-precision activation $\mathcal{A}_f$, and α, β coefficients. At L, EmMark compares the extracted weight $\mathcal{W}'[L]$ with the original weight $\mathcal{W}[L]$ and gets their difference $\Delta\mathcal{W}[L]$ using Equation 6. Model owners can assert ownership by comparing $\Delta\mathcal{W}[L]$ with inserted signature sequence B.

$$\Delta\mathcal{W}[L] = \mathcal{W}'[L] - \mathcal{W}[L] \tag{6}$$

The watermark extraction rates are computed using Equation 7, where $|B|$ is the length of the inserted signature, and $|B|'$ is the number of matching signature bits.

$$\%WER = 100 \times \frac{|B|'}{|B|} \tag{7}$$

Watermarking strength. Equation 8 evaluates the probability that a non-watermarked model matches the inserted signatures by chance. k is the number of matching bits between the owner's and non-watermarked model's signatures. $|B|$ is the signature length. The signature generation follows the Rademacher distribution, and each bit has an equal probability of 0.5 to be 1 or -1.

$$P_c = \sum_{i=k}^{|B|} \left(\begin{array}{c} |B| \\ i \end{array} \right) 0.5^{|B|} \tag{8}$$

5 EXPERIMENTS

5.1 Experiment Setup

Target Model We use LLaMA-2 [21] and OPT [27] family models with parameter sizes ranging from 125 million to 70 billion as the target LLM. The models are compressed and quantized by Smoothquant [24] to INT8 for OPT family, by LLM.int8() [5] to INT8 for LLaMA-2 family, and by AWQ [12] to INT4. Note that EmMark is agnostic to quantization algorithms. We use the the frameworks as a proof-of-concept showing EmMark's performance. We watermark on top of the official model quantization instances [23].
Watermark Parameters For the target model, we insert 300-bit signatures per INT8 quantized layer and 40-bit per INT4 quantized layer. It yields a minimum watermarking strength of 9.09×10^{-13} for each layer, and 9.09×10^{-13n} for n quantized layer LLM following Equation 8. Such watermarking strength provides sufficient protection to the target models. The coefficients α and β are set to 0.5 and 0.5. The seed for signature insertion is 100. For model size smaller than 6.7 billion, $\frac{|B_c| \times n}{|B|} = 50$. For larger LLMs, $\frac{|B_c| \times n}{|B|} = 60$.
Baselines We compare EmMark with RandomWM where watermarks are inserted in random indexes and SpecMark where watermarks are inserted into the high-frequency region transformed by DCT. SpecMark is designed for full precision models, and we apply the transformation to the quantized weights.
Evaluation Metrics The watermarked LLM performance is evaluated using *Perplexity (PPL)* for text fluency on WikiText Dataset [14], and, *Zero-shot Accuracy (Zero-shot Acc)* for token prediction on the mean of LAMBADA, HellaSwag, PIQA, and WinoGrande Datasets [8]; the watermark extraction performance is evaluated using the percentage of signatures successfully extracted as *Watermark Extraction Rate (WER)*; the watermark efficiency is evaluated using the *Insertion Time (Time)* and required *GPU Memory (Memory)*.

5.2 Results

EmMark's Fidelity. Different watermarking frameworks' performance is in Table 1. The first and the second cells are the perplexity and zero-shot accuracy degradation for accommodating the signatures. The third cell is the watermark extraction rates.
Comparison with RandomWM: While RandomWM performs on par with the non-watermarked models in INT8 quantization, it failed to preserve lower-bit quantization models quality. Its performance drops significantly in INT4 quantization, where the degradation is 2.29 and 0.23%, respectively, for perplexity and zero-shot accuracy. As a result, the watermarked LLM has worse text fluency.

In contrast, EmMark introduced no performance degradations compared with non-watermarked LLMs in both INT8 and INT4 quantization. By strategically scoring quantized parameters, EmMark preserved the model quality during watermark insertion.

Comparison with SpecMark: As seen from Table 1, SpecMark failed to watermark embedded LLM and achieved 0% WER in both INT8 and INT4 quantization. The weight distributions in embedded LLMs are sparse, and the small additions/deletions from the DCT domain cannot change the discrete parameters in the weight domain. As a result, SpecMark failed to watermark such LLMs. EmMark, however, successfully inserted signatures into the embedded model without introducing additional quality deterioration.

EmMark's Efficiency. The efficiency is evaluated by the required time (Time) and GPU memory (Memory) for watermark insertion on the OPT family models [27]. In Table 2, we report the average time/memory for signature insertion per quantization layer. For both INT8 and INT4 quantization, the time taken to watermark one quantization layer is less than 0.4s. The insertion takes less than 10 minutes for the largest OPT-30B model. All of EmMark's components are performed on CPUs, and no additional GPU memory is required for watermark insertion. As such, the efficient and lightweight watermarking scheme makes EmMark scalable to LLMs with larger parameter sizes.

Quantization	Time (s)	Memory (GB)
INT8	0.4	0
INT4	0.3	0

Table 2: EmMark's watermarking efficiency.

5.3 Robustness

We evaluate EmMark's robustness by performing the following attacks: (i) parameter overwriting attacks, (ii) re-watermark attacks, and (iii) forging attacks. We use the OPT-2.7B [27] model quantized to INT4 by AWQ [12] as the target model.
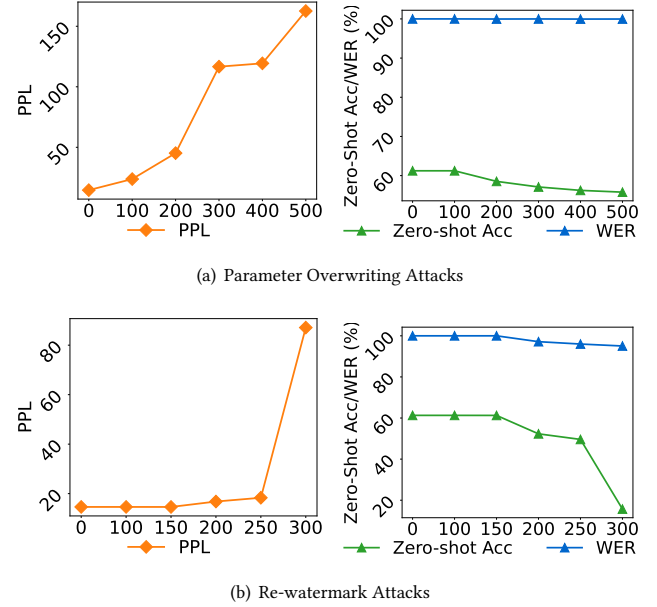
The embedded model is already compressed and quantized. Therefore, additional pruning attacks will result in model ability breakdown. Current fine-tuning on the quantized models uses LoRA-based approaches [6] to add low-rank adaptors to learn new dataset features. Such methods, however, will not change the quantized model weight parameters, and cannot be used to remove signatures.

Parameter Overwriting Attacks. The adversary removes the watermark by randomly adding one bit to the parameter weights in the watermarked model. The attacked model performance (perplexity and zero-shot accuracy) and watermark extraction rates are shown in Figure 2(a). The number of overwritten parameters in each quantized layer is increased from 100 to 500 with a constant gap of 100. As seen, the attacked model performance drops as more bits are overwritten. The threshold renders significant model quality degradations is at 300-bit, where the PPL is over 100 and indicates bad generated text fluency. Nevertheless, EmMark demonstrates its robustness under such attacks and maintains over 99% WER.

Re-watermark Attacks. The adversary knows EmMark's general watermark insertion algorithm. However, he/she cannot access the model owners' signatures or random seeds. The adversary tries to break the watermark by perturbing parameters potentially used for watermarking. The watermark coefficients α and β and seed

differ from the watermark insertion stage and are set to 1, 1.5, and 22, respectively. The activation for scoring S_r is obtained from the quantized LLM instead of the full-precision one. The quantization model performance (perplexity and zero-shot accuracy) and watermark extraction rates are shown in Figure 2(b). The number of perturbed parameters in each quantized layer is increased from 100 to 300 with a constant gap of 50.

Figure 2(b) shows EmMark maintains its high watermark extraction rates under re-watermark attacks. The threshold for bad LLM quality is at 300 bits, where the zero-shot accuracy is below 20% and yields bad token prediction performance. However, the watermarked model still maintains over 95% watermark extraction rates, providing sufficient IP protection to the embedded LLM.

(a) Parameter Overwriting Attacks

(b) Re-watermark Attacks

Figure 2: EmMark's performance under parameter overwriting and re-watermarking attacks. The left subplot evaluates Perplexity (PPL), and the right subplot depicts Zero-shot Accuracy and Watermark Extraction Rate (WER).

Forging Attacks. The adversary does not remove the LLM owners' watermark. Instead, he/she claims the model ownership by faking another set of watermarks. It is achieved by (i) counterfeiting the watermark weight locations L_a with a fake signature sequence; (ii) re-watermarking on top of the watermarked embedded LLM by a counterfeited full-precision model activations and insertion hyperparameters. EmMark is resilient to the two settings with a confidential full-precision model's activation that the adversary does not have access to.

In the first setting, the watermark weight locations are a reproduction of the full-precision model's activations and a set of hyperparameters (random seed and scoring coefficients). Counterfeiting the location L_a without such reproduction validation process leads to failed forging attacks. In the second setting, while the adversary proves ownership of the re-watermarked model, the inserted signatures from owners are still highly extractable as seen from the **Re-watermark Attacks** section. Notably, matching the owners' signatures by coincidence is nearly impossible and the

probability goes as low as 9.09×10^{-13n} for n quantization layer LLM following Equation 8. Here, n=192 for OPT-2.7B, and larger as the model size grows. As such, the owners are able to claim ownership under forging attacks and protect the IP.

5.4 Ablation Study and Analysis

We provide further ablation study and analysis of EmMark's capabilities. The OPT-2.7B [27] quantized by AWQ [12] to INT4 is the target model.

Effectiveness of Watermark Coefficients. We analyze how different α and β choices affect EmMark's watermarking performance in Table 3. α is changed from 0 to 1, and β is changed from 1 to 0. The maximum watermark signature length is set to 100. As shown, as the coefficient β becomes larger, EmMark tends to choose bits in the saliency channel over bits with larger values. While such insertion ensures strong IP protection, the watermarked LLM's quality is compromised.

(α, β)	PPL $\downarrow$	Zero-shot Acc (%) $\uparrow$	WER (%) $\uparrow$
(1, 0)	14.61	61.36	100
(0.5, 0.5)	14.61	61.36	100
(0, 1)	14.65	61.25	100

Table 3: Effectiveness of different insertion coefficients.

Watermark Capacities. We show the maximum signature length can be inserted into the embedded LLM without compromising its quality in Figure 3. We increase the inserted length per quantization layer from 50 to 200 with a constant gap of 50. As seen, the threshold for maintaining watermarked LLM performance is at 100 bit. It corresponds to a watermark strength of 1.57×10^{-30} per layer and 1.57×10^{-5760} for OPT-2.7B model following Equation 8, providing sufficient protection to the model IP.
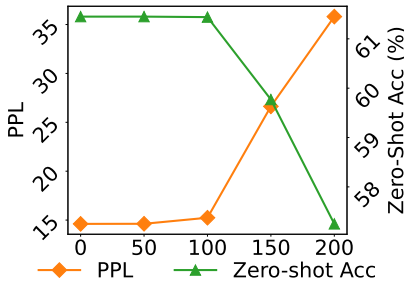

Figure 3: EmMark's watermark performance with inserted signature lengths increased from 50-bit to 200-bit. All of the watermarks are successfully extracted.

Watermark Integrity. To ensure integrity, EmMark shall only prove ownership of watermarked models and generate low WER on non-watermarked ones. In Table 4, the first model is the non-watermarked OPT-2.7B quantized by AWQ [12], the second model is fine-tuned on a 4k subset of the Alpaca dataset [20] before AWQ [12] quantization, the third model is fine-tuned on Wiki-Text dataset [14] before AWQ [12] quantization, and the fourth model is non-watermarked OPT-2.7B quantized by GPTQ [7]. The EmMark successfully extracts all signatures from the watermarked model and fails to prove ownership of non-watermarked ones. As a result, EmMark demonstrates its integrity.

Model	WM	non-WM 1	non-WM 2	non-WM 3	non-WM 4
WER (%)	100	0	0	0	0

Table 4: EmMark's integrity evaluation on watermarked and non-watermarked models.

6 CONCLUSION

We present EmMark, a novel watermarking framework for embedded large language models. It inserts signatures into the compressed and quantized models while demonstrating quality preservation and robustness. As such, it tackles the security challenges arising from deploying large language models in edge devices and protects the IP of model owners. Extensive evaluations on OPT and LLaMA-2 family models show EmMark successfully inserts 100% signatures into the LLM with no performance compromises.

REFERENCES

[1] Franziska Boenisch. 2021. A systematic review on model watermarking for neural networks. *Frontiers in big Data* 4 (2021), 729663.
[2] Huili Chen et al. 2019. Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In *ICMR*. 105–113.
[3] Huili Chen et al. 2020. SpecMark: A Spectral Watermarking Framework for IP Protection of Speech Recognition Systems.. In *INTERSPEECH*. 2312–2316.
[4] Bita Darvish Rouhani et al. 2019. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *ASPLOS*.
[5] Tim Dettmers et al. 2022. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339* (2022).
[6] Tim Dettmers et al. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314* (2023).
[7] Elias Frantar et al. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323* (2022).
[8] Leo Gao et al. 2021. *A framework for few-shot language model evaluation*. https://doi.org/10.5281/zenodo.5371628
[9] Mojan Javaheripi et al. 2022. LiteTransformerSearch: Training-free Neural Architecture Search for Efficient Language Models. *NeurIPS* 35 (2022), 24254–24267.
[10] Linyang Li et al. 2023. Watermarking LLMs with Weight Quantization. *arXiv preprint arXiv:2310.11237* (2023).
[11] Yiming Li et al. 2022. Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection. *NeurIPS* 35 (2022), 13238–13250.
[12] Ji Lin et al. 2023. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. *arXiv preprint arXiv:2306.00978* (2023).
[13] Xinyin Ma et al. 2023. LLM-Pruner: On the Structural Pruning of Large Language Models. *arXiv preprint arXiv:2305.11627* (2023).
[14] Stephen Merity et al. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843* (2016).
[15] Qualcomm. 2022. Whitepaper: The future of AI is hybrid. *Qualcomm blog* (2022).
[16] John Schulman et al. 2022. ChatGPT: Optimizing language models for dialogue. *OpenAI blog* (2022).
[17] Masoumeh Shafieinejad et al. 2021. On the robustness of backdoor-based watermarking in deep neural networks. In *ACM IH&MMSec workshop*. 177–188.
[18] David So et al. 2021. Searching for efficient transformers for language modeling. *NeurIPS* 34 (2021), 6010–6022.
[19] Mingjie Sun et al. 2023. A Simple and Effective Pruning Approach for Large Language Models. *arXiv preprint arXiv:2306.11695* (2023).
[20] Rohan Taori et al. 2023. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.
[21] Hugo Touvron et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).
[22] Xiuying Wei et al. 2022. Outlier suppression: Pushing the limit of low-bit transformer language models. *NeurIPS* 35 (2022), 17402–17414.
[23] Thomas Wolf et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).
[24] Guangxuan Xiao et al. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *ICML*. 38087–38099.
[25] Runxin Xu et al. 2022. From dense to sparse: Contrastive pruning for better pre-trained language model compression. In *AAAI*, Vol. 36. 11547–11555.
[26] Zhewei Yao et al. 2023. A comprehensive study on post-training quantization for large language models. *arXiv preprint arXiv:2303.08302* (2023).
[27] Susan Zhang et al. 2022. OPT: Open Pre-trained Transformer Language Models. arXiv:2205.01068