Domain-Specific LLM Adaptation: Bridging Personalization and Efficiency Through Synthetic Data and Optimization

Iman Abbasnejad[1], **Brett Tully**[2], **Wei Zhou**[2], **Tomal Deb**[1], **Sheldon Liu**[1], **Xuefeng Liu**[1], **Warren Wei**[1]

[1]Amazon Web Service, [2]Nearmap

{imanaba, tomalde, shilong, liuxuefe, warrewei}@amazon.com, {brett.tully, wei.zhou}@nearmap.com

Abstract

Large Language Models (LLMs) have demonstrated exceptional capabilities but face two critical deployment challenges: high computational costs and scarcity of personalized domain training data. We address these dual challenges through a comprehensive framework that combines synthetic data generation with inference optimization techniques. Our approach employs LLMs for zero-shot and few-shot synthetic dataset creation while applying structural pruning, knowledge distillation, quantization, and prompt caching for computational efficiency. We evaluate three architectural paradigms: encoder-only, encoder-decoder, and decoder-only on synthetic building permit classification and assess optimization techniques on public benchmarks. Our systematic evaluation reveals that strategic architectural selection based on task characteristics is more critical than model complexity: encoder-only models provide superior efficiency-accuracy trade-offs for high-throughput scenarios, demonstrating that understanding problem requirements enables effective deployment without requiring the largest available models. Knowledge distillation emerges as the key optimization technique for personalized domains, recovering pruning-induced performance degradation with Llama3-8B+KD achieving 87.1% accuracy at 20% pruning while exceeding unpruned baselines. Complementary optimization strategies, including dynamic caching (5-18% latency reduction with zero performance loss) and hardware acceleration (up to 200× speedup) enable flexible deployment configurations for domain specific applications.

Introduction

LLMs (Touvron et al. 2023; Thoppilan et al. 2022; Le Scao et al. 2023) have demonstrated remarkable capabilities through extensive pre-training (Brown et al. 2020; Chowdhery et al. 2023). Their exceptional performance on complex language tasks has driven widespread adoption despite their substantial computational requirements compared to more compact NLP models such as BERT (Devlin et al. 2019) and MicroBERT (Gessler and Zeldes 2022). As these architectures expand, they manifest emergent abilities (Wei et al. 2022) while concurrently presenting two significant challenges: substantial computational demands for deployment and limited availability of high-quality personalized domain data for fine-tuning. The enormous data requirements of contemporary models consuming hundreds of billions to trillions of tokens during pre-training make adaptation particularly challenging in personalized domains where labeled data is sparse or unavailable.

Our research addresses these parallel challenges through a comprehensive approach. First, we investigate techniques for data-efficient fine-tuning by utilizing existing LLMs to generate synthetic training datasets for personalized domains. This synthetic data generation strategy provides a promising solution to the data scarcity issue, facilitating domain adaptation without requiring extensive manually-labeled examples. Second, we explore diverse inference optimization methodologies to enable cost-effective deployment, including structural pruning (Filters'Importance 2016; Wang, Wohlwend, and Lei 2019), knowledge distillation (Sun et al. 2019, 2020), and quantization techniques (Frantar et al. 2022; Bai et al. 2020).

In this work, we perform a systematic evaluation of both synthetic data generation approaches for LLMs and inference optimization strategies. Our analysis spans multiple dimensions, from conventional performance metrics (accuracy, perplexity, ROUGE-L) to practical deployment considerations such as inference latency across different hardware configurations. By simultaneously addressing computational efficiency and data scarcity challenges, our research provides actionable solutions for deploying LLM technologies in personalized domains where both computational resources and domain-specific training examples are limited.

Related works

The optimization and deployment of large language models (LLMs) faces two critical challenges that our research addresses: obtaining sufficient high-quality training data and achieving efficient inference in resource-constrained environments. These challenges are deeply interconnected, as both stem from the fundamental characteristics of modern LLMs and their deployment requirements.

Traditional fine-tuning approaches for domain adaptation require substantial personalized domain examples, which are often unavailable or prohibitively expensive to obtain. Research has explored various alternatives to address this data scarcity problem, including few-shot learning (Brown et al. 2020), prompt engineering (Schulhoff et al. 2024), and

parameter-efficient fine-tuning methods like LoRA (Hu et al. 2022) and prefix tuning (Li and Liang 2021). While these techniques reduce data dependencies to varying degrees, they often struggle with personalized domains requiring deep domain knowledge.

The emergence of LLMs has instigated a significant paradigm shift in addressing both challenges (Zhang et al. 2023; Bang et al. 2023). Despite their impressive capabilities, high-quality data remains the foundation for building robust NLP models (Gandhi et al.), while the computational demands of these models stemming from their extensive pretraining on vast corpora (Brown et al. 2020; Touvron et al. 2023), create deployment barriers that require innovative optimization approaches.

An emerging paradigm that addresses the data scarcity challenge is using LLMs themselves to generate synthetic training data (Long et al. 2024; Xu et al. 2023). This bootstrapping technique leverages knowledge embedded in pretrained models to create personalized domain examples for fine-tuning. Recent work has shown promising results using synthetic data generation across diverse tasks including classification (Meng et al. 2022), instruction tuning (Wang et al. 2021; Honovich et al. 2022; Wang et al. 2023), mathematics (Yu et al. 2023; Luo et al. 2023a), and code generation (Luo et al. 2023b). Popular models like Alpaca (Taori et al. 2023), Vicuna (Zheng et al. 2023), and Openchat 3.5 (Wang et al. 2023) demonstrate that synthetic data has become instrumental in developing high-performance yet efficient language models.

This synthetic data approach offers advantages beyond addressing scarcity. Several studies (Hosking, Blunsom, and Bartolo 2023; Gilardi, Alizadeh, and Kubli 2023) have highlighted that human-generated data, being inherently susceptible to biases and errors, may not be optimal for model training. Synthetic data can be designed to overcome these limitations, though creating datasets with both high correctness and sufficient diversity requires careful process design and numerous technical considerations (Gandhi et al.).

Complementary to data generation, inference optimization techniques have evolved to address deployment efficiency. Model pruning represents a foundational approach, ranging from simple magnitude-based methods (Han et al. 2015; Zafrir et al. 2021) to sophisticated approaches using first-order importance estimation (Hou et al. 2020) and hessian-based calculations (Kurtic et al. 2022; Xu et al. 2021; Liu et al. 2021). Structural pruning (Wang, Wohlwend, and Lei 2019; Xia, Zhong, and Chen 2022) is particularly relevant for deployment scenarios, creating smaller but dense models that maintain hardware acceleration benefits.

Knowledge distillation (Sun et al. 2019, 2020) creates a natural bridge between our two focus areas, enabling smaller models to leverage knowledge extracted from larger teacher models (Ma et al. 2022; Rashid et al. 2020) while simultaneously addressing efficiency concerns. This technique allows knowledge transfer without requiring access to the vast datasets used to train the teacher models, making it synergistic with synthetic data generation approaches. Quantization methods (Bai et al. 2020; Yao et al. 2022) further complement these techniques by reducing numerical precision requirements while preserving core capabilities.

Recent innovations have accelerated these optimization processes through post-training compression techniques (Kwon et al. 2022; Frantar and Alistarh 2023; Abbasnejad, Deb, and Liu 2025) and layer-wise approaches (Sun et al. 2023; Frantar et al. 2023; Xia et al. 2023; Guo et al. 2023) that apply pruning techniques sequentially for greater efficiency. These advances create opportunities for more integrated approaches to model optimization.

Our work uniquely addresses these dual challenges by systematically evaluating how synthetic data generation techniques interact with inference optimization methods. This integrated perspective is crucial because optimization strategies that work well with human-annotated data may perform differently when applied to models fine-tuned on synthetic data. By examining these interactions, we provide a comprehensive framework for deploying LLMs in resource-constrained environments and personalized domains, offering practical solutions for organizations facing constraints in both computational resources and personalized domain training data availability.

Method

In this section, we present our framework for personalized domain data generation for model training alongside techniques for evaluating and optimizing LLM during inference.

Dataset overview

Our generated data focuses on the classification of building permit data within a personalized domain context, specifically predicting the likelihood of roof-related changes. The dataset consists of synthetic building permit records containing fields such as project type, description, and permit status. Each permit record includes information about construction or renovation projects across residential and commercial domains. The classification task involves determining whether each permit relates to changes in roof structures, assigning a confidence score (0-100) that quantifies the certainty of roof modification. High confidence scores (70-100) indicate clear roof-related work such as direct roofing projects or new construction with explicit roof components. Medium confidence scores (40-69) suggest potential roof impact through major structural modifications or complete demolitions. Low confidence scores (0-39) represent permits unlikely to involve roof changes, such as interior renovations.

Synthetic Data Generation Our data generation framework utilizes the inherent knowledge and generative capabilities of LLMs to create personalized domain training data without requiring extensive human-labeled examples. We implement two distinct generation paradigms: zero-shot and few-shot generation.

In the zero-shot setting, we employ personalized prompting techniques that establish task boundaries and domain specifications without supplying explicit exemplars. The LLM leverages its comprehensive pre-trained knowledge base to synthesize contextually appropriate samples that conform to the stipulated requirements for the personalized domain. Following initial generation, a human evaluator con-

ducts quality assessment of the produced samples, verifying their adherence to personalized domain conventions and task objectives. This human-in-the-loop validation step ensures that generated content maintains semantic integrity and personalized domain-specific structural patterns. The validation process encompasses verification of both content appropriateness and format compliance, establishing a quality control mechanism that guarantees consistency throughout the synthetic dataset. Through this hybrid automated-human approach, we maintain high standards of data fidelity while scaling beyond the limitations of purely manual data creation methodologies. Once several high-quality samples are generated and validated, we use them as reference points for the LLM for few-shot data generation tailored to the personalized domain.

Data Labeling The next step of our personalized domain data generation framework is labeling the generated data. The labeling system incorporates a hierarchical scoring mechanism that evaluates multiple dimensions of each permit: (1) project type analysis, which examines primary classification indicators within the personalized domain context; (2) description analysis, which evaluates the detail level and specificity of work described according to personalized domain conventions; and (3) supporting evidence assessment, which considers additional contextual factors relevant to the personalized domain. This multi-dimensional evaluation generates confidence scores between 0-100 which is used as the likelihood of the roof change.

Model Optimization

In this section, we present our framework for various inference optimization techniques for LLMs specialized for personalized domains. Our method includes five key approaches: model pruning, knowledge distillation, quantization methods, prompt caching, and hardware acceleration through AWS Inferentia.

Model pruning We use the layer-pruning method from (Gromov et al.), which is based on the observation that representations in transformer networks evolve gradually across layers. In a transformer architecture, each layer follows a residual iteration:

$$x^{(\ell+1)} = x^{(\ell)} + f(x^{(\ell)}, \theta^{(\ell)}) \tag{1}$$

where $x^{(\ell)}$ and $\theta^{(\ell)}$ represent the multi-dimensional input and parameter vectors for layer ℓ, and $f(x, \theta)$ describes the transformation of a multi-head self-attention and MLP layer block. The pruning algorithm identifies layers where representations remain similar across multiple layers, making those intermediate layers potentially redundant. It computes the angular distance between inputs at different layers and removes the layers where this distance is minimized. This is based on the hypothesis that if representations change slowly such that $x^{(\ell)} \approx x^{(\ell-1)} + \epsilon$ with $\epsilon \ll x^{(\ell)}$, then certain layers can be removed with minimal impact on performance. After pruning the redundant layers, fine-tuning can be applied to heal any representation mismatches, with particular attention to preserving performance on personalized domain tasks.

Knowledge Distillation This approach (Hinton, Vinyals, and Dean 2015) is a neural network compression technique where a smaller "student" model learns to mimic the behavior of a larger, more complex "teacher" model, especially valuable for personalized domain applications. In the process, the student model captures the generalized knowledge embedded within the teacher's learned representations rather than just the hard class labels. This can be done by training the student model to match the teacher's output probability distributions, which are typically "softened" using a temperature parameter, T in the softmax function. The softened distributions (where $T > 1$) reveal more information about the teacher's internal representations than hard labels alone, showing not just which class the teacher predicts but the relative similarities between classes that the teacher has learned. In practice, distillation for personalized domains often uses a weighted combination of two objective functions: matching the teacher's soft probability distributions and predicting the correct hard labels, with the soft targets usually given higher priority. This approach effectively transfers the teacher's generalized knowledge while maintaining task performance in the personalized domain, enabling significantly smaller models to achieve comparable accuracy to their larger counterparts.

Model Quantization Quantization reduces the numerical precision of model weights and activations, decreasing memory requirements and computations while attempting to preserve model behavior for personalized domain applications. Neural network computations typically use high-precision floating-point formats (FP32) during training, but this level of precision is often unnecessary during inference. By reducing precision, we can achieve efficiency gains with minimal impact on model quality. We evaluate multiple quantization methods across various precision levels including, FP32, BF16, INT8 and INT4.

Prompt Caching Prompt caching aims to avoid redundant computation by storing and reusing results from previously processed prompts, particularly effective in personalized domain applications where similar queries are frequently encountered. The KV (Key-Value) cache optimizes autoregressive generation by storing previously computed attention vectors, $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$, reducing computational complexity from $O(t^2)$ to $O(t)$ for a sequence of length t, though requiring memory that scales as $2 \times L \times h \times d \times t$ bytes for a model with L layers, h attention heads, and hidden dimension d. In this work we use three different prompt caching techniques: (i) static caching, (ii) dynamic Semantic Caching and (iii) HQQ (Hierarchical Quantized Query) caching, which combines quantized vector representations with hierarchical indexing to dramatically reduce memory requirements while maintaining rapid retrieval of KV values.

Experiments

We conduct comprehensive experiments across two distinct scenarios to evaluate our dual approach of synthetic data generation and inference optimization. Our experimental framework addresses both data scarcity scenarios through

synthetic data generation and computational efficiency challenges through various optimization techniques.

Experimental Setup

Hardware Configuration: All experiments were conducted on NVIDIA A100 GPUs with 2×40GB memory configuration for quantization experiments, while AWS Inferentia2 instances (1 chip, 32 cores, 32 NeuronCores, 128GB memory) were used for hardware acceleration evaluation.

Model Selection: We evaluate three model families with varying parameter scales: Llama3 (1B, 8B), Qwen2.5 (32B), representing different computational complexity tiers to assess scalability across resource constraints.

Dataset: We evaluate our approach on two complementary datasets addressing different deployment scenarios. To assess model performance and inference optimization in low-data environments, we created a dataset of building permit records for predicting roof-related changes, comprising 2,000 training samples, 300 validation samples, and 200 test samples, all generated using our zero-shot and few-shot synthetic data generation approach explained in section . Each permit record contains structured information including project type, description, and permit status, with the task being to predict confidence scores (0-100) indicating likelihood of roof modifications. To evaluate inference optimization techniques on public benchmarks, we use the Unnatural Instructions dataset (Honovich et al. 2022) containing 68k text instructions, formatted using templates from (Wang et al. 2022), with a validation set of 1,000 examples for model selection. Testing is performed on LMentry (Efrat, Honovich, and Levy 2022) and Super-Natural Instructions (Wang et al. 2022) benchmarks.

Model Architecture Comparison: We evaluate models across different architectural paradigms and parameter scales. For low-data scenarios in personalized domains, we systematically compare three architectural families across 13 models: (1) Decoder-Only Models (Llama 3.2-1B, Llama 3-8B, Qwen2-1.5B, Qwen2-0.5B) that perform both confidence score prediction and reasoning generation in a single pass with conversational formatting, (2) Encoder Models (BERT variants) adapted for classification by binning confidence scores into discrete ranges with classification heads, and (3) Encoder-Decoder Models (T5 series) treating the task as text-to-text transformation with "predict roof score:" prefixes. For inference optimization evaluation, we focus on three foundation models as baselines: Llama3-1B, Llama3-8B (Grattafiori et al. 2024) and Qwen2.5-32B (Yang et al. 2024; Team 2024), representing different computational complexity tiers to assess scalability across resource constraints.

Optimization Techniques: Our optimization framework encompasses multiple approaches. We implement knowledge distillation where Qwen2.5-32B serves as the teacher model and smaller Llama models serve as students. Model pruning follows the methodology from (Gromov et al.), focusing on identifying and removing redundant components while minimizing performance impact. We also investigate combined approaches where knowledge distillation is performed on pruned variants, using the full Qwen2.5-32B model as teacher and pruned Llama models as students, aiming to recover performance lost during pruning while maintaining efficiency benefits. Additional optimization techniques include prompt caching (dynamic, HQQ, and static variants), quantization across multiple precision levels (FP32, BF16, INT8, INT4), and hardware acceleration through AWS Inferentia2.

Evaluation Metrics: Our evaluation includes both performance and efficiency for each experimental scenario. For synthetic data generation in personalized domain settings, we assess performance using Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Exact Match rate, accuracy within 5 points, accuracy within 10 points, and inference time for 200 test samples. For inference optimization on public benchmarks, we use inference time (seconds per sample and tokens per second), perplexity (measuring uncertainty in next token prediction), ROUGE-L (measuring overlap between generated and reference text), LMentry scores (Efrat, Honovich, and Levy 2022) (using greedy decoding with high-accuracy regular expressions), exact match rates, and accuracy metrics.

Results

We performed an extensive series of evaluations on the proposed techniques and compare the performance and computational complexity of the methods across both experimental scenarios.

Limited Data Experiment Table 1 presents comprehensive performance comparison across architectural paradigms on our synthetic building permit dataset. Encoder-decoder models demonstrated superior performance overall, with T5 Large achieving the best results across most metrics (MAE=2.30, RMSE=6.26, 94% accuracy within 10 points). The T5 family's text-to-text framework proved particularly well-suited for numerical prediction tasks requiring precise outputs, making it ideal for confidence score predictions.

Encoder-only models showed remarkable efficiency despite their simpler architecture, with BERT Base achieving strong performance (MAE=3.30, 79.5% exact match) while maintaining the fastest inference time (6.25 seconds for 200 samples). This suggests that for this specific task of classifying building permits in a personalized domain, the classification-oriented approach of encoder models is highly effective and computationally efficient, offering an excellent efficiency-to-accuracy ratio.

Decoder-only models generally underperformed on numerical prediction tasks, with even Llama 3B achieving only 5.05 MAE, though they provided valuable reasoning capabilities unavailable in other architectures. The performance degradation was particularly severe in smaller decoder-only models like Qwen 0.5B (MAE=31.34), showing steep performance decline with size reduction from Llama 3B (MAE=5.05) to Qwen 0.5B (MAE=31.34). This suggests that for domains requiring precise numerical predictions, traditional encoder-decoder architectures may still hold advantages over decoder-only models despite the broader success of LLMs in general language tasks.

Inference Optimization Experiment We now present results of inference optimization techniques on public dataset.

Table 1: Performance comparison across model architectures on synthetic building permit dataset.

Model	MAE↓	RMSE↓	EM↑	W5↑	W10↑	Time↓
Encoder-Only Models						
BERT Base	3.30	8.77	79.5%	86.0%	89.5%	6.25
BERT Large	3.65	9.95	77.0%	85.0%	89.5%	7.38
RoBERTa Base	3.55	7.94	68.5%	82.5%	92.5%	6.22
RoBERTa Large	4.10	9.14	68.0%	82.0%	89.5%	7.34
Encoder-Decoder Models						
T5 Large	2.30	6.26	80.0%	89.0%	94.0%	31.41
T5 Base	3.38	9.41	78.0%	86.0%	90.0%	13.68
T5 Small	4.33	9.60	68.5%	80.5%	88.0%	8.15
Flan T5 Large	3.62	7.95	68.5%	82.5%	90.5%	14.87
Flan T5 Base	3.75	8.29	69.5%	81.5%	90.0%	14.66
Flan T5 Small	4.30	9.80	69.5%	82.6%	88.0%	8.70
Decoder-Only Models						
Llama 3B	5.05	11.43	68.5%	80.0%	86.0%	76.56
Llama 1B	7.05	13.11	51.5%	71.0%	81.0%	37.58
Qwen 1.5B	7.80	15.77	48.0%	52.5%	61.0%	71.20
Qwen 0.5B	31.34	39.53	9.0%	14.0%	20.0%	39.92

Table 2: LMentry benchmark results with latency improvements.

Model	Prun.	Cache	Performance		Efficiency		
			Perp.↓	LMent↑	Lat.↓(s)	Tok/s↑	Imp.(%)
Lla3-1B	0%	None	15.38	59.2	2.42	1067	-
Lla3-1B	0%	Dyn	15.38	59.2	1.99	1272	17.8%
Lla3-1B	20%	Dyn	19.85	51.8	1.83	1445	24.4%
Lla3-1B	40%	Dyn	26.19	43.6	1.53	1633	36.8%
Lla3-1B+KD	0%	None	13.92	61.7	2.32	1077	-
Lla3-1B+KD	0%	Dyn	13.92	61.7	1.97	1268	15.1%
Lla3-1B+KD	20%	Dyn	17.47	56.3	1.81	1434	22.0%
Lla3-1B+KD	40%	Dyn	22.64	49.4	1.51	1551	34.9%
Lla3-8B	0%	None	10.32	60.1	2.91	844	-
Lla3-8B	0%	Dyn	10.32	60.1	2.53	1010	13.1%
Lla3-8B	20%	Dyn	14.78	55.6	2.33	1152	19.9%
Lla3-8B	40%	Dyn	20.92	47.5	1.92	1343	34.0%
Lla3-8B+KD	0%	None	9.18	63.5	2.89	879	-
Lla3-8B+KD	0%	Dyn	9.18	63.5	2.42	1030	16.3%
Lla3-8B+KD	20%	Dyn	12.97	59.8	2.25	1163	22.1%
Lla3-8B+KD	40%	Dyn	18.26	52.9	1.94	1311	32.9%
Qwen-32B	0%	None	11.54	65.9	3.74	673	-
Qwen-32B	0%	Dyn	11.54	65.9	3.53	795	5.6%
Qwen-32B	20%	Dyn	15.37	56.4	2.79	918	25.4%
Qwen-32B	40%	Dyn	22.11	48.7	2.34	1125	37.4%

Base Model Performance: Our analysis begins with the baseline performance of each model without pruning, caching, or knowledge distillation. As shown in Table 2, Llama3-1B achieved an LMentry score of 59.2 with a latency of $2.42s$, while Llama3-8B reached 60.1 points at $2.91s$, and Qwen2.5-32B attained 65.9 points with the longest latency of $3.74s$. On the Super-Natural Instructions benchmark, Table 3, we observe a clearer performance hierarchy, with Qwen2.5-32B achieving 27.8% exact match (EM) and 91.7% accuracy, followed by Llama3-8B (23.5% EM, 84.9% accuracy) and Llama3-1B (18.7% EM, 78.3% accuracy). These results establish our performance baseline for subsequent optimization experiments.

Model Pruning Impact: When applying model pruning, we observe a consistent trade-off between computational efficiency and task performance. At 20% pruning with no other optimizations, Llama3-1B's LMentry score decreased to 51.8 while improving latency to $1.83s$ (24.4% reduction). Similar patterns emerge for Llama3-8B (55.6) and Qwen2.5-32B (56.4), with latency improvements of 19.9% and 25.4%, respectively.

At 40% pruning, more substantial performance degradation occurs, Llama3-1B's LMentry score falls to 43.6, Llama3-8B to 47.5, and Qwen2.5-32B to 48.7. However, the latency improved by 36.8%, 34.0%, and 37.4%, respectively. Notably, larger models appear more resilient to aggressive pruning, maintaining better relative performance at higher pruning rates.

On the Super-Natural Instructions benchmark, Table 3, 20% pruning reduced exact match rates by 2.9 (to 15.8%), 2.8 (to 20.7%), and 3.3 (to 24.5%) percentage for Llama3-1B, Llama3-8B, and Qwen2.5-32B respectively. At 40% pruning, this degradation increased to 6.1 (to 12.6%), 7.1 (to 16.4%), and 8.2 (to 19.6%) percentage. The corresponding decreases in accuracy follow a similar pattern, with reductions of 4.7%, 3.6%, and 3.7% respectively. From this analysis, we observe that the percentage-wise reduction in exact match rate is more severe than the reduction in accuracy, suggesting that pruning primarily affects the model's precision in generating exact outputs for domain specific tasks rather than its overall understanding of instructions.

Caching Strategies: For personalized domain applications where similar queries may frequently recur, caching strategies offer pure efficiency gains with minimal performance impact. As shown in Table 2, dynamic caching provides substantial latency improvements across all models: 17.8% for Llama3-1B ($2.42s$ to $1.99s$), 13.0% for Llama3-8B ($2.91s$ to $2.53s$), and 5.6% for Qwen2.5-32B ($3.74s$ to $3.53s$). Importantly, these latency reductions come with virtually no decline in performance metrics, preserving identical LMentry scores, exact match rates, and accuracy percentages.

HQQ caching offers varied improvements: 7.4% for Llama3-1B, 10.3% for Llama3-8B, and 8.0% for Qwen2.5-32B. This strategy incurs minimal performance penalties (0.1 to 0.2 on LMentry), making it a viable alternative when memory efficiency is prioritized in domain specific deployments. Static caching provides the most modest efficiency gains of 4.1 to 7.6% across models, with Qwen2.5-32B showing only 3.5% improvement, but maintains performance parity with the no-cache baseline.

When analyzing caching strategies across pruned models for specific domains, we observe that the relative efficiency

Table 3: Super-Natural Instructions results with EM rate changes. Best accuracy highlighted (pink). EM $\Delta(\%)$ shows change relative to baseline (0% pruning, no KD).

Model	Pruning	EM(%)	Acc(%)	EM Δ
Llama-1B	0%	18.7	78.3	-
Llama-1B	20%	15.8	73.6	-15.5%
Llama-1B	40%	12.6	68.2	-32.6%
Llama-1B+KD	0%	21.3	83.7	+13.9%
Llama-1B+KD	20%	19.1	80.6	+2.1%
Llama-1B+KD	40%	15.8	74.5	-15.5%
Llama-8B	0%	23.5	84.9	-
Llama-8B	20%	20.7	81.3	-11.9%
Llama-8B	40%	16.4	75.8	-30.2%
Llama-8B+KD	0%	26.7	89.5	+13.6%
Llama-8B+KD	20%	24.3	87.1	+3.4%
Llama-8B+KD	40%	20.2	82.4	-14.0%
Qwen-32B	0%	27.8	91.7	-
Qwen-32B	20%	23.5	88.0	-15.5%
Qwen-32B	40%	19.6	82.1	-29.5%

Table 4: Inference times across quantization methods. INT4 Impr. shows latency reduction vs. baseline; Inf. Impr. shows Inferentia speedup.

Model	Pruning	FP32	INT4	Inf.	INT4↓	Inf.↑
Llama-1B	-	2.42	0.94	0.012	61%	201×
Llama-1B	20%	1.93	0.96	0.007	50%	276×
Llama-8B	-	2.91	0.91	0.043	69%	68×
Llama-8B	20%	2.05	1.98	0.037	3%	55×
Qwen2.5B-32B	-	3.74	2.30	0.872	39%	4×
Qwen2.5B-32B	20%	3.65	3.66	0.641	0%	6×

gains remain consistent regardless of pruning level. Caching strategies consistently provide additional latency gains regardless of pruning level, with dynamic caching reducing latency by $15-18\%$ for Llama3 models and $5-7\%$ for Qwen2.5-32B across different pruning rates. This suggests that caching and pruning offer complementary optimization benefits in personalized domain applications.

Knowledge Distillation: Knowledge distillation proves particularly valuable for personalized domain applications, where transferring domain expertise from larger to smaller models is essential. Unpruned Llama3-1B+KD achieved an LMentry score of 61.7 (from 59.2), approaching the performance of the much larger Llama3-8B. Similarly, Llama3-8B+KD reached 63.5 from 60.1. On the Super-Natural Instructions benchmark, knowledge distillation yielded significant gains in exact match rates: from 18.7% to 21.3% for Llama3-1B and from 23.5% to 26.7% for Llama3-8B. This brings the Llama3-8B+KD model to within 1.1% of Qwen2.5-32B's performance despite being four times smaller.

The most significant benefit of knowledge distillation for personalized domains appears when combined with pruning. At 20% pruning, Llama3-1B+KD achieved an LMentry score of 56.3, recovering 4.5 points of the 7.4-point loss from pruning. Similarly, Llama3-8B+KD reached 59.8, recovering 4.2 points and nearly matching the unpruned base Llama3-8B. Similar patterns emerge on the Super-Natural Instructions benchmark, where knowledge distillation at 20% pruning improved EM from 15.8% to 19.1% for Llama3-1B and from 20.7% to 24.3% for Llama3-8B. Notably, Llama3-8B+KD with 20% pruning achieved 87.1% accuracy, higher than Llama3-8B with no pruning (84.9%) and approaching Qwen2.5-32B with 20% pruning (88.0%).

Even at 40% pruning, distilled models maintained substantially better performance in personalized domain tasks: Llama3-1B+KD scored 49.4 on LMentry (vs. 43.6 without KD) and Llama3-8B+KD scored 52.9 (vs. 47.5 without KD), demonstrating that knowledge distillation can mitigate performance degradation even at higher pruning ratios in specialized domains.

The most significant benefit of knowledge distillation appears when combined with pruning. At 20% pruning, Llama3-1B+KD achieved an LMentry score of 56.3, recovering 4.5 points of the 7.4-point loss from pruning. Similarly, Llama3-8B+KD reached 59.8, recovering 4.2 points and nearly matching the unpruned base Llama3-8B. Similar pattern is observed on the Super-Natural Instructions benchmark, where knowledge distillation at 20% pruning improved EM from 15.8% to 19.1% for Llama3-1B and from 20.7% to 24.3% for Llama3-8B. Notably, Llama-8B+KD with 20% pruning achieved 87.1% accuracy, higher than Llama3-8B with no pruning (84.9%) and approaching Qwen2.5-32B with 20% pruning (88.0%). Even at 40% pruning, distilled models maintained substantially better performance: Llama3-1B+KD scored 49.4 on LMentry (vs. 43.6 without KD) and Llama-8B+KD scored 52.9 (vs. 47.5 without KD), demonstrating that knowledge distillation can mitigate performance degradation even at higher pruning ratios.

Hardware Acceleration and Quantization: Table 4 presents inference time across different quantization methods and hardware platforms. All quantization experiments were conducted on NVIDIA A100 GPUs with $2\times$40GB GPUs. Quantization alone provides significant acceleration, with INT4 quantization reducing latency by 61.2% for Llama3-1B (from $2.42s$ to $0.94s$), 68.7% for Llama3-8B (from $2.91s$ to $0.91s$), and 38.5% for Qwen2.5-32B (from $3.74s$ to $2.30s$) compared to unquantized baselines. However, the most dramatic improvements come from hardware acceleration with AWS Inferentia2, which delivers $201.7\times$ speedup for Llama3-1B (from $2.42s$ to $0.012s$), $67.7\times$ for Llama3-8B (from $2.91s$ to $0.043s$), and $4.3\times$ for Qwen2.5-32B (from $3.74s$ to $0.872s$). When combining Inferentia with model pruning for personalized domain applications, we observe additional performance gains, with pruned models achieving latency reductions of 41.7% (Llama3-1B), 14.0% (Llama3-8B), and 26.5% (Qwen2.5-32B), highlighting the complementary benefits of specialized hardware and model optimization techniques.

Conclusion

In this work, we addressed the dual challenges of data scarcity and computational efficiency that hinder practical LLM deployment in personalized domains. Our comprehensive eval-

uation demonstrates that when personalized domain data is limited, synthetic data generation provides a viable pathway to achieving high-accuracy models through strategic architectural selection. Our experiments on building permit classification reveal that encoder-decoder models, particularly T5 Large, achieves superior performance (MAE=2.30, 94% accuracy within 10 points) on synthetically generated datasets, while encoder-only models like BERT Base offer optimal efficiency-accuracy trade-offs for production deployment in these specialized contexts. Concurrently, our evaluation of inference optimization techniques establishes knowledge distillation as the most effective approach for maintaining personalized domain performance while reducing computational requirements, with distilled models achieving over 90% of teacher performance despite 4× parameter reduction. By demonstrating that synthetic data generation can overcome training data limitations in personalized domains while layered optimization techniques dramatically improve inference efficiency, our framework provides practical solutions for deploying LLM technologies in specialized contexts where both high-quality domain-specific training data and computational resources are constrained. These findings have significant implications for organizations seeking to leverage LLM capabilities in resource-constrained environments while maintaining high performance on personalized domain tasks.

References

Abbasnejad, I.; Deb, T.; and Liu, X. 2025. Maximizing LLM Efficiency Through Optimization Strategies. In *KDD 2025 Workshop on Inference Optimization for Generative AI*.

Bai, H.; Zhang, W.; Hou, L.; Shang, L.; Jin, J.; Jiang, X.; Liu, Q.; Lyu, M.; and King, I. 2020. Binarybert: Pushing the limit of bert quantization. *arXiv preprint arXiv:2012.15701*.

Bang, Y.; Cahyawijaya, S.; Lee, N.; Dai, W.; Su, D.; Wilie, B.; Lovenia, H.; Ji, Z.; Yu, T.; Chung, W.; et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H. W.; Sutton, C.; Gehrmann, S.; et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 4171–4186.

Efrat, A.; Honovich, O.; and Levy, O. 2022. Lmentry: A language model benchmark of elementary language tasks. *arXiv preprint arXiv:2211.02069*.

Filters'Importance, D. 2016. Pruning Filters for Efficient ConvNets.

Frantar, E.; and Alistarh, D. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, 10323–10337. PMLR.

Frantar, E.; Ashkboos, S.; Hoefler, T.; and Alistarh, D. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Frantar, E.; Ashkboos, S.; Hoefler, T.; and Alistarh, D.-A. 2023. OPTQ: Accurate post-training quantization for generative pre-trained transformers. In *11th International Conference on Learning Representations*.

Gandhi, S.; Gala, R.; Viswanathan, V.; Wu, T.; and Neubig, G. ???? Better synthetic data by retrieving and transforming existing datasets, 2024b. *URL https://arxiv. org/abs/2404*, 14361.

Gessler, L.; and Zeldes, A. 2022. MicroBERT: Effective Training of Low-resource Monolingual BERTs through Parameter Reduction and Multitask Learning. In *Proceedings of the The 2nd Workshop on Multi-lingual Representation Learning (MRL)*, 86–99. Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics.

Gilardi, F.; Alizadeh, M.; and Kubli, M. 2023. ChatGPT outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30): e2305016120.

Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Gromov, A.; Tirumala, K.; Shapourian, H.; Glorioso, P.; and Roberts, D. A. ???? The unreasonable ineffectiveness of the deeper layers, 2024. *URL https://arxiv. org/abs/2403.17887*.

Guo, S.; Xu, J.; Zhang, L. L.; and Yang, M. 2023. Compresso: Structured pruning with collaborative prompting learns compact large language models. *arXiv preprint arXiv:2310.05015*.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Honovich, O.; Scialom, T.; Levy, O.; and Schick, T. 2022. Unnatural instructions: Tuning language models with (almost) no human labor. *arXiv preprint arXiv:2212.09689*.

Hosking, T.; Blunsom, P.; and Bartolo, M. 2023. Human feedback is not gold standard. *arXiv preprint arXiv:2309.16349*.

Hou, L.; Huang, Z.; Shang, L.; Jiang, X.; Chen, X.; and Liu, Q. 2020. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33: 9782–9793.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2): 3.

Kurtic, E.; Campos, D.; Nguyen, T.; Frantar, E.; Kurtz, M.; Fineran, B.; Goin, M.; and Alistarh, D. 2022. The optimal bert surgeon: Scalable and accurate second-order pruning for large language models. *arXiv preprint arXiv:2203.07259*.

Kwon, W.; Kim, S.; Mahoney, M. W.; Hassoun, J.; Keutzer, K.; and Gholami, A. 2022. A fast post-training pruning framework for transformers. *Advances in Neural Information Processing Systems*, 35: 24101–24116.

Le Scao, T.; Fan, A.; Akiki, C.; Pavlick, E.; Ilić, S.; Hesslow, D.; Castagné, R.; Luccioni, A. S.; Yvon, F.; Gallé, M.; et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Li, X. L.; and Liang, P. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Liu, Z.; Li, F.; Li, G.; and Cheng, J. 2021. EBERT: Efficient BERT inference with dynamic structured pruning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 4814–4823.

Long, L.; Wang, R.; Xiao, R.; Zhao, J.; Ding, X.; Chen, G.; and Wang, H. 2024. On llms-driven synthetic data generation, curation, and evaluation: A survey. *arXiv preprint arXiv:2406.15126*.

Luo, H.; Sun, Q.; Xu, C.; Zhao, P.; Lou, J.; Tao, C.; Geng, X.; Lin, Q.; Chen, S.; and Zhang, D. 2023a. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Luo, Z.; Xu, C.; Zhao, P.; Sun, Q.; Geng, X.; Hu, W.; Tao, C.; Ma, J.; Lin, Q.; and Jiang, D. 2023b. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*.

Ma, X.; Wang, X.; Fang, G.; Shen, Y.; and Lu, W. 2022. Prompting to distill: Boosting data-free knowledge distillation via reinforced prompt. *arXiv preprint arXiv:2205.07523*.

Meng, Y.; Huang, J.; Zhang, Y.; and Han, J. 2022. Generating training data with language models: Towards zero-shot language understanding. *Advances in Neural Information Processing Systems*, 35: 462–477.

Rashid, A.; Lioutas, V.; Ghaddar, A.; and Rezagholizadeh, M. 2020. Towards zero-shot knowledge distillation for natural language processing. *arXiv preprint arXiv:2012.15495*.

Schulhoff, S.; Ilie, M.; Balepur, N.; Kahadze, K.; Liu, A.; Si, C.; Li, Y.; Gupta, A.; Han, H.; Schulhoff, S.; et al. 2024. The prompt report: a systematic survey of prompt engineering techniques. *arXiv preprint arXiv:2406.06608*.

Sun, M.; Liu, Z.; Bair, A.; and Kolter, J. Z. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Sun, S.; Cheng, Y.; Gan, Z.; and Liu, J. 2019. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*.

Sun, S.; Gan, Z.; Cheng, Y.; Fang, Y.; Wang, S.; and Liu, J. 2020. Contrastive distillation on intermediate representations for language model compression. *arXiv preprint arXiv:2009.14167*.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Stanford alpaca: An instruction-following llama model.

Team, Q. 2024. Qwen2.5: A Party of Foundation Models.

Thoppilan, R.; De Freitas, D.; Hall, J.; Shazeer, N.; Kulshreshtha, A.; Cheng, H.-T.; Jin, A.; Bos, T.; Baker, L.; Du, Y.; et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Wang, G.; Cheng, S.; Zhan, X.; Li, X.; Song, S.; and Liu, Y. 2023. Openchat: Advancing open-source language models with mixed-quality data. *arXiv preprint arXiv:2309.11235*.

Wang, S.; Liu, Y.; Xu, Y.; Zhu, C.; and Zeng, M. 2021. Want to reduce labeling cost? GPT-3 can help. *arXiv preprint arXiv:2108.13487*.

Wang, Y.; Mishra, S.; Alipoormolabashi, P.; Kordi, Y.; Mirzaei, A.; Arunkumar, A.; Ashok, A.; Dhanasekaran, A. S.; Naik, A.; Stap, D.; et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

Wang, Z.; Wohlwend, J.; and Lei, T. 2019. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*.

Wei, J.; Tay, Y.; Bommasani, R.; Raffel, C.; Zoph, B.; Borgeaud, S.; Yogatama, D.; Bosma, M.; Zhou, D.; Metzler, D.; et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Xia, M.; Gao, T.; Zeng, Z.; and Chen, D. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.

Xia, M.; Zhong, Z.; and Chen, D. 2022. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*.

Xu, D.; Yen, I. E.; Zhao, J.; and Xiao, Z. 2021. Rethinking Network Pruning–under the Pre-train and Fine-tune Paradigm. *arXiv preprint arXiv:2104.08682*.

Xu, R.; Cui, H.; Yu, Y.; Kan, X.; Shi, W.; Zhuang, Y.; Jin, W.; Ho, J.; and Yang, C. 2023. Knowledge-infused prompting: Assessing and advancing clinical text data generation with large language models. *arXiv preprint arXiv:2311.00287*.

Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B.; Zhou, C.; Li, C.; Li, C.; Liu, D.; Huang, F.; Dong, G.; Wei, H.; Lin, H.; Tang, J.; Wang, J.; Yang, J.; Tu, J.; Zhang, J.; Ma, J.; Xu, J.; Zhou, J.; Bai, J.; He, J.; Lin, J.; Dang, K.; Lu, K.; Chen, K.; Yang, K.; Li, M.; Xue, M.; Ni, N.; Zhang, P.; Wang, P.; Peng, R.; Men, R.; Gao, R.; Lin, R.; Wang, S.; Bai, S.; Tan, S.; Zhu, T.; Li, T.; Liu, T.; Ge, W.; Deng, X.; Zhou, X.; Ren, X.; Zhang, X.; Wei, X.; Ren, X.; Fan, Y.; Yao, Y.; Zhang, Y.; Wan, Y.; Chu, Y.; Liu, Y.; Cui, Z.; Zhang, Z.; and Fan, Z. 2024. Qwen2 Technical Report. *arXiv preprint arXiv:2407.10671*.

Yao, Z.; Yazdani Aminabadi, R.; Zhang, M.; Wu, X.; Li, C.; and He, Y. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35: 27168–27183.

Yu, L.; Jiang, W.; Shi, H.; Yu, J.; Liu, Z.; Zhang, Y.; Kwok, J. T.; Li, Z.; Weller, A.; and Liu, W. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Zafrir, O.; Larey, A.; Boudoukh, G.; Shen, H.; and Wasserblat, M. 2021. Prune once for all: Sparse pre-trained language models. *arXiv preprint arXiv:2111.05754*.

Zhang, C.; Zhang, C.; Zheng, S.; Qiao, Y.; Li, C.; Zhang, M.; Dam, S. K.; Thwal, C. M.; Tun, Y. L.; Huy, L. L.; et al. 2023. A complete survey on generative ai (aigc): Is chatgpt from gpt-4 to gpt-5 all you need? *arXiv preprint arXiv:2303.11717*.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36: 46595–46623.