Energy-Efficient Inference with Small Language Models: A Comparative Study on Code Generation, Classification, and Environmental Impact

Anonymous authors
Paper under double-blind review

Abstract

Large language models (LLMs) are widespread in enterprise applications for code completion, email classification, and sentiment analysis. Although these models have good performance, their high computational requirements make them consume high energy in inference. Can smaller language models (SLMs) with three billion parameters (Qwen2.5-3B-Instruct) perform similarly in structured high-frequency tasks while providing significantly lower environmental impact?

We tested an SLM on three enterprise workloads: code generation with HumanEval benchmark (164 tasks), HR email routing (1,339 examples), and binary sentiment analysis (100 samples). We recorded output quality, inference latency, throughput, GPU memory utilisation and energy consumption. The SLM achieved 72.6% pass rate on code generation and 86% on sentiment analysis at 388–647× less energy per query than GPT-4o on code tasks and 210–1,333× less on classification tasks. Scaling to organisational context, replacing LLMs with task-specific SLMs for 10,000 daily code completions, 100,000 sentiment queries, and 50,000 monthly email classifications would save 52,642 kWh annually, reducing CO2 emissions by 23.1 metric tons. An SLM-first deployment strategy is a practical way to attain sustainable AI with significant energy savings.

1 Introduction

Huge language models also penetrated the software development process and enterprise automation with remarkable speed. Code completion with aid of AI is used by developers, whereas such models find application in organisations to route emails, carry out sentiment analysis, and classify documents. There are common features of these applications; they are continuous, take large amounts of incoming requests, and generally produce succinct but compact output.

Recent works have done little beyond investigating the energy cost of model training and concluded that large model requires considerable computational resources to train. However, the inference energy usage is the biggest part of the total lifecycle footprint after the deployment of the models (Jegham et al., 2025). The inference operations are linear with any number of execution and query, compared with training in which inference is performed only after one time. Per-request inefficiencies that are minor at any rate can scale to huge environmental and operational expenditures since a single organisation can handle the inference requests (often millions per day).

This scenario brings a basic mismatch: general-purpose models are used in most high-frequency enterprise work, despite any simple and deterministic required outputs. Activities like those of deriving functions based on specifications, sorting emails in given categories or even the binary sentiment classification take up a narrow output space and are very predictable. The characteristics of these models imply that task-specific models with orders of magnitude fewer parameters can support similar performance levels while achieving significant reduction in computational cost.

The primary question the present work is built on is whether it is possible to achieve parity in the quality of small language models in the context of structured and repeatable tasks and achieve a considerable cutback in the amount of energy that is used during the inference. This would confirm an SLM-first implementation paradigm: small models would be deployed by default and only when more problem complexity or ambiguity require more capacity should bigger models be increased.

2 Problem Statement

The modern-day use of AI seems ineffective: organisations utilize large-scale general-purpose language models to solve the problems that do not justify the entire capacity of the model or the ongoing computational cost. In this paper, a study on the use of small language models (around three billion parameters) to facilitate common enterprise workloads and produce orders of magnitude improvements on inference energy costs is explored.

There are three representative types of tasks whose evaluation is performed:

Code Generation: The developers would demand an implementation of the functions depending on the explicit demands and test them against the predefined unit tests. Such jobs require syntactic and logical accuracy under a clearly defined group of constraints.

Email Classification: The enterprise system classifies incoming traffic to a set type by either department or priority level. Space used for labeling is fixed and predictions must be singleton.

Sentiment Analysis: The sentiment of customer review, products and surveys are categorized into binary or ternary customer sentiment. The encoding into discrete groups produced by such activities does not mean that text is generated in a simple manner.

Each of the three types has traits that are similar to the abilities of smaller models: structured inputs, constrained outputs, low semantic ambiguity and high query frequency. A transition into smaller models can turn out to be quite practical and even useful when it comes to the environmental and economic impact of performing such tasks on the organisational level.

3 Contributions

This manuscript contributes to the existing body of literature on the topic of energy efficient deployment of language models in the following ways:

1. **Empirical Study of SLM Performance:** We verify and lower bound performance of a 3-billion parameter language model execution on a set of three overall enterprise tasks and we measure both quality indicators (Pass@1 for code and classification accuracy) and resource indicators (latency, throughput, memory usage).

2. **Energy Consumption Analysis:** We evaluate the energy-efficiency improvement that the deployment of the small language model (SLM) offers in comparison to the modern large language model (LLM), comparing the amount of energy required to run the small language model at work against the current large language model, and generalizing the results to the organizational scale.

3. **Task Suitability Framework:** We identify task properties that define the feasibility of the SLM e.g., output organization, length constraints and entropy and offer deployment decision guidelines.

4. **Environmental Impact Estimation:** We estimate the tangible energy savings, and CO2 reductions which can be achieved under SLM-first deployment strategies and prove the instant sustainability rates with the absence of any infrastructural modifications.

5. **Hybrid Deployment Strategy:** We suggest an escalation procedure, where the SLM is involved in managing standard queries and shifting ambiguous or difficult queries to bigger models, which is optimal in terms of the accuracy versus efficiency trade-off.

6. **Reproducibility:** We have supportive replication, indicating that we give fully trained empirical models, available in Google Colab notebooks that can be replicated and altered to fit other situations.

4 Related Work

4.1 Code Generation Benchmarks

HumanEval benchmark proposed by Chen et al. (2021) developed a standardized structure of testing python programming synthesis skills against 164 problems along with sources of unit tests. Pass@k measure was used which is the likelihood of at least one program sampled passing tests. Later benchmarks such as MBPP (Austin et al., 2021) and APPS (Hendrycks et al., 2021) broadened both the scale diversity and language diversity, and difficulty hypotheses. Most of the studies that have been conducted here focused heavily on functional correctness without paying much attention to computation efficiency. Recent publications (Ashraf et al., 2025) have bridged this gap by quantifying energy consumption as well as accuracy by proving what smaller models that can make correct outputs always use less energy than larger ones, hence achieving improvements in efficiency without compromising quality.

4.2 Small Language Models

Small language models that are traditionally defined as having less than ten billion parameters have become effective replacements of large general-purpose models in task-specific contexts. Models like Qwen2.5-Coder, CodeLlama variants, and Phi family are also able to deliver a competitive performance on the targeted tasks but do so consuming significantly fewer resources in terms of computational resources (ArXCompass, 2024). Research on SLMs concentrates more on domain specific applications, demonstrating that with proper fine-tuning or prompt engineering, such models can be able to be equivalent and at times superior to larger systems in specialized tasks. However, the systematic cross-task comparisons especially on the use of energy among various workloads are under-researched.

4.3 Enterprise Automation through NLP

The functions implemented in enterprise NLP include email filtering, spam detection, and sentiment analysis. The classical machine-learning methods, such as gradient-boosted trees and Naïve Bayes, have been compared to transformer-based systems and sentence transformers. Big models have performed well in few-shot, as well as, low-resource conditions (Labonne & Moran, 2023). In particular comparisons of GPT-4 with domain-specific models including BERT and XLNet document that smaller model types can equal or out-perform LLMs on less general data, like social media text, and in considerably cheaper circumstances (Zhang et al., 2023). The above observations highlight the high impact of model adaptation on predictive performance and efficiency.

4.4 Sustainability and Energy of AI System

AI systems have also made the environmental footprint and energy footprint critical concerns when it comes to large-scale deployment. Studies are being done to quantify the energy, water and carbon footprint of inference of LLM inference in different configurations. It has been shown that the output of a single query to a commercial LLM can use many watt-hours, and combined usage can be disastrous at scale (Jegham et al., 2025). Although typically most studies on sustainability are training costs, inference energy, in many cases, will be the most prominent lifecycle consumption of deployed systems (Fernandez et al., 2025). However, the existing literature on systematic task-level comparisons of energy use when optimising a deployed system to be sustainable is limited, and this is a major gap in knowledge. The current paper has outlined this shortcoming by presenting task-specific energy comparative curves of small and large models and organizational scale of impact projections, thus offering evidence-based decision supports in selection of models in production settings where sustainability and operational costs are the major factor.

5 Methodology

5.1 Model Selection

The Qwen2.5-3B-Instruct model which is used in the current evaluation was selected based on its instruction-oriented nature which is friendly to prompt-following tasks and its balance of capacity and efficiency. This model has about three billion parameters and therefore, falls within the size regime, which is ideal in exploring utility on routine tasks in the enterprise and what performance improvement can be achieved as compared to the larger systems.

To optimize the use of the resources, we used 4-bit quantization through the BitsAndBytes library. This quantization has minimal effects on output quality and decreases the amount of memory consumed on the GPU and inference latency when doing classification and short-form generation tasks. The criteria that were used to select the model:

Instruction-tuning compatibility: Appropriateness toward specification of workload based on prompt to allow easy comparison

Open availability: The power to reproducibility and practicability

Representative size: Parameter count should match small language models that can be deployed to support selected areas of task

5.2 Experimental Infrastructure

The experiments were all run on Google Colab Pro which provides accessibility and can be replicated as it runs using cloud resources common in accessibility. The hardware setup was composed of NVIDIA T4 GPU having 16 GB of memory, which is typical of portable mid-range accelerators used in the cloud. These conditions are realistic business constraints as opposed to specialized high-performance infrastructure.

Specifications of software environment:

- Python 3.10+
- PyTorch 2.x
- Transformers 4.x
- BitsAndBytes (4-bit quantization)
- Hugging Face Datasets

Version controlled code with model loading, quantization parameters, batching and evaluation metrics is available as Google Colab notebooks linked in the abstract. GPU memory usage was measured with the help of PyTorch utilities which reported the peak memory consumption at the point of model execution and just after the call. Monitoring of CPU memory was done using psutil. Measures of latency only include the forward pass time, but not the data loading, tokenization, or after-processing overhead.

5.3 Tasks and Datasets

The current study on the SLM was based on three main categories of tasks which characterize the workload within a typical enterprise environment. These activities include generative analysis of structured texts and categorization of them that accurately represent real-life deployment conditions and validated measurement criteria. Table 1 outlines the experimental design.

Code Generation: HumanEval benchmark contains 164 synthesis problems in Python containing unit tests. Pass@1 measures the percentage of functions generated which pass all the test cases in the first attempt hence giving an idea of the functional correctness.

Table 1: Tasks, datasets, and evaluation metrics used in our experiments

Task	Dataset	Primary Metric
Code Generation	HumanEval (164 tasks)	Pass@1
Email Classification	HR Routing (1,339 examples)	Accuracy
Sentiment Analysis	Product Reviews (100 samples)	Accuracy

Email Classification: Our email corpus is a manually maintained enterprise email routing and labeling project. The data set has 1,339 labeled examples, each one is in one category of routing. Accuracy is used to refer to the ratio of the correctly classified categories to the ground truth.

Sentiment Analysis: The standard corpus of product reviews with binary positive or negative labels, which contains 100 reviews, is used by us. The accuracy is determined whereby the percentage of the predictions (sentiments) matches the reference labels.

5.4 Evaluation Metrics

The evaluation scheme determines the quality of the output and the amount of resources that were consumed with every type of task. All of the metrics that are reflected in the given tables are the result of a number of test runs.

Quality Metrics:

Pass@1 (code generation): This is the probability that instead of failing all unit tests, the generated solution passes all unit tests the first time it is run

Accuracy (classification): Percentage of accurateness on the classification compared to the ground-truth labels

Measures of Resource Usage:

Latency (seconds): End to end inference wall-clock time of a request

Throughput (tokens/s): This is a rate of token generation in a sequence-to-sequence model

Peak GPU Memory (MB): The maximum rate of GPU memory usage over inference

Energy Proxy: This is the approximate energy consumption that is calculated as latency and peak memory product and it is used when exact power measurements are not available (Fernandez et al., 2025)

6 Results

All the results present here are based on experimental executions that were carried in the relative Google Colab notebooks.

6.1 Quality Evaluation

The effectiveness of the SLM on the three task categories based on the benchmark measure is listed in Table 2.

In the case of code generation on HumanEval, SLM has a pass rate of 72.6%, which means that approximately three of the four functions pass the unit tests on the first attempt. This is a competitive result of a 3-billion-parameter model, specifically in comparison with the increase in the cost of the computational cost of larger ones.

Table 2: Task quality results from our experimental evaluation

Task	Metric	Value (%)
Code Generation	Pass@1	72.6
Email Classification	Accuracy	71.0
Sentiment Classification	Accuracy	86.0

Table 3: Resource utilization measurements from our experiments

Task	Latency (s)	Tokens/s	Peak GPU (MB)
Code Generation	9.60	8.87	2,080
Email Classification	1.57	2.22	2,138
Sentiment Classification	0.73	3.10	2,079

Sentiment classification was also able to classify the product-review test set with 86% accuracy which is strong performance in simple binary classification tasks. The binary form of sentiment cues makes the models very accurate, even in smaller models.

The classification of email was found to be more difficult whereby the SLM recorded 71% accuracy on the data used in HR routing. These findings highlight disparity in classes and poor representation in specific groups which could be improved by balancing the dataset or fine-tuning tasks.

6.2 Performance and Resource Utilization

Table 3 tells the resources-usage measurements of every activity giving evidence of the connection among the traits and computational requirements of tasks.

Task latency is significantly different among tasks, and is influenced mostly by length of output requirements. Code generation has the highest latency as autoregressive token generation of multi-line functions incurs higher latency compared to classification tasks which only use a single forward pass.

The highest used GPU memory consumption is also relatively low with an average of 2.1 GB across the tasks which can be explained by 4-bit quantization. The area of model size and quantization level are the primary factors that affect the memory performance regardless of the input properties or type of tasks thus are able to enable predictable allocation of resources to production deployments.

Data on resource-utilization which has been realized show that the inference cost is pre-eminently also decided by the length of the output resource and complexity of the generation process since the computational overhead of tasks with prolonged generation processes is proportionally greater than a task with a single-token classification resource.

6.3 Error Analysis

The analysis of failure modes explains the shortcomings of the SLM and guides the right strategy of escalation.

Code Generation Failures: Numerous faulty solutions are made on issues that require complex nesting logic or complex algorithms or processing of edge-case peculiarities. Similar to syntactically correct code generated by the SLM, semantic correctness is also more often than not, demonstrating a limitation beyond which models of larger size or manual checking are required.

Email Classification Errors: Misclassifications are focused on doubtful cases and those categories which are underrepresented. The label confusion comes about when the message content cuts across more than one category or there is no definite departmental signifier. The balancing of the datasets and the inclusion of more contextual data can help in addressing them.

Table 4: Per-query energy consumption comparison from our experiments

Task	**LLM (Wh)**	**SLM (Wh)**	**Reduction**
Code Generation	9.71	0.015–0.025	388–647×
Email Classification	4.0	0.003–0.005	800–1,333×
Sentiment Classification	0.42	0.001–0.002	210–420×

Table 5: Annual influence of SLM deployment predicted according to our experiments

Workflow	**Queries**	**Energy (kWh)**	**CO_2 (t)**
Code Generation	10,000 daily	35,369	15.5
Sentiment Analysis	100,000 daily	15,275	6.7
Email Classification	50,000/month	2,398	1.05
Total	–	52,642	23.1

Sentiment Analysis Errors: The mixed-sentiment and sarcastic language as well as subtle context cues do not work favorably with the SLM. Decision-making on clean positive or negative examples is nearly accurate but diminishes when the sentiment information is not on the surface but involves some inference. This limitation can be alleviated through targeted fine-tuning on ambiguous target samples that are task specific.

On the whole, the error analysis confirms the conclusion that SLMs are good at well-defined tasks without ambiguity, complexity or under-represented patterns, and that they need an escalation mechanism in new situations.

7 Energy Usage and Impact on Environment

7.1 Per-Query Energy Comparison

Literature in the recent past cites inference energy consumption as one of the key contributors to the environmental footprint through the entire life cycle of deployed AI systems. The models like GPT-4o have been estimated by commercial studies to use approximately 0.42 Wh per query, and the power usage directly depends on query length and model size (Jegham et al., 2025).

We use latency × peak GPU memory as a proxy of energy in the case that we do not have direct measurements of power in our experiments. This measure is highly correlated with actual power expenditure in company of uniform hardware setups and it allows the comparison of models of different size and task profiles.

Our experimental results provide estimates of the per-query energy of large language models and SLM. Table 4 gives the results.

On conservative estimates small language models (SLM) can reduce energy usage by two to three orders of magnitude compared to large general purpose models (on per query basis). This very significant saving is due mainly to the much smaller number of parameters and to the shortness of the generation sequences for SLMs, both factors having a direct impact on computational requirements.

7.2 Organizational-Scale Impact

It can be seen that the deployment of SLMs has a measurable environmental impact at the organizational level upon considering the per query efficiencies. Table 5 quantifies annual energy as well as CO2 emission reductions, which are based on realistic query loads in an enterprise context and experimentally determined energy ratios.

Table 6: Task characteristics and suitability measured in our experiments

Task	Structure	Length	Entropy	Performance
Code Gen.	Deterministic	Short-Med	Moderate	72.6% Pass@1
Email Class.	Fixed labels	Single token	Low	71.0% accuracy
Sentiment	Binary	Single token	Very low	86.0% accuracy

The energy savings calculations are based on the baseline energy spending of large language models (LLMs) and realistic data on query distributions of mid-sized organizations. CO2 emission reductions were modelled by a standard grid electric carbon intensity of 0.438 kg CO2/kWh (conservative estimates with relevance to a geography).

The results show that the redistribution of high frequency routine operation from large models to SLMs can save considerable annual energy, exceeding 52,000 kWh in a single enterprise with an average consumption of AI. This amounts to not emitting 23 metric tons of CO2, which is the same amount as leaving around five passenger vehicles parked for one year.

7.3 Scaling Characteristics

A number of important insights come out of the energy analysis:

Linear scaling with query volume: Energy use shows a linear relationship with request rate; hence, efficiency gains on high frequency requests may be reflected in significant absolute savings in millions of requests per day.

Inference dominance: Previous evidence has shown that the energy costs for inference are significantly higher than the energy costs for training models destined for production environments (Fernandez et al., 2025). Therefore, optimizing the inference processes is the main approach to reducing the environmental impact of Artificial Intelligence (AI) deployment.

Potential for immediate deployment: An SLM-first deployment paradigm offers immediate benefits in terms of sustainability without requiring new architecture designs or hardware upgrades for the same. Current models and infrastructure can support speedy implementation.

These analyses validate the idea that task specific model selection is a practical and high impact intervention for Artificial Intelligence that is sustainable. By reserving model capacity proportional to actual task requirements, organizations can make orders-of-magnitude energy savings with no achieving task compromise.

8 Discussion

8.1 Suitability of a Task for Small Language Models

Evaluation in different categories of tasks shows quite definite patterns of suitability for SLMs. Table 6 presents task characteristics and SLM performance measured in our experiments.

Tasks that are characterised by structured inputs, constrained outputs and low semantic entropy have been consistently achieved with high SLM performance. In contexts like this, large general purpose models offer marginal benefits in comparison to the increased computational price.

Sentiment analysis is a perfect SLM scenario as a result of the fact that it is a binary text classification problem with explicit surface-level cues. Likewise, email routing has a fixed label space with low ambiguity making it amenable for the SLMs.

8.2 Constraints and Triggers of Escalation

SLMs face observed limitations for escalation to larger ones:

Complicated multi-step reasoning: Problems involving code generation which require nested logic, difficult algorithms or unusual edge cases are often too challenging for SLMs.

Under-represented categories: Severe class imbalance or rare labels deteriorate the performance of classification because of the low exposure to training.

Contextual ambiguity: Tasks in which one must interpret distorted meaning of sarcasm, mixed signals or sensitive cues add a burden to the representational power of smaller models.

In the HumanEval benchmark, 119 out of 164 problems were solved successfully by the SLM with a 72.6% success rate. Failures were due not to language modeling inferiority but to a lack in the capacity to handle given algorithmic complexities and edge cases. This finding lends credence to a hybrid approach of having SLMs combat the standard scenarios and then defer the more complex or uncertain scenarios to larger models.

8.3 Utilization Patterns for Resources

Analysis of energy consumption suggests that output length is the main factor in inference cost. Tasks that have single-token responses have minimal latency; generative tasks have costs that are linear in sequence length. This relationship does not depend on model size, and instead suggests that production cost is determined mainly by the nature of the task, rather than model choice.

Peak GPU memory usage is not coupled to the nature of the task for which it operates thanks to quantization, and latency, rather than memory becomes the main factor in energy consumption as well in fixed hardware setups. As a result, it is essential that the maximum energy savings by shortening the output time and batching take priority over the strategies to improve memory.

8.4 Deployment Strategy

Empirical findings outline a prescription of a tripartite deployment framework:

Use SLMs for structured and large number of tasks as a default: When things are limited in output and specifications are defined narrowly, it is important to use small models as the main inference backend.

Utilize confidence-based escalation: Low confidence or out of distribution instances must be directed to larger models or even to humans.

Match the capacity of models to task uncertainty: Model selection should represent semantic complexity and ambiguity, and not a generic perceived task importance.

This paradigm reverses the traditional deployment logic where maximum capability and/or power usually takes precedence over more immediate need to allow organizations to realize immediate gain in efficiency without architectural modification or retraining, through task centric model selection.

9 Limitations and Future Work

Several of the limitations of the present study warrant acknowledgement:

Energy Measurement methodology: Direct measurements of power were not available, and so proxy measures were used. While proven to be accurate in previous work, absolute values of energy can be affected by the hardware configurations. Future research should combine the direct power measurement through hardware monitoring equipment.

Task coverage: Only three archetypal task types were reviewed, of course enterprise AI has a more expansive scope. Extend evaluation to other loads of work (question answering, summarisation and translation) for more generalisation.

Model selection: The research was focused on one 3-billion-parameter model. Systematic evaluation across SLM sizes (1B, 3B, 7B) would help shed more light on the trade-offs associated with scaling.

Dataset scale: Email classification and sentiment analysis data sets were small, which may not allow it to reflect the diversity of production. More extensive corpora will allow good performance estimates.

The future research directions include:

Automated routing systems that are dynamically driven by input characteristics and confidence estimates to route models based on them

Fine-tuning strategies that are tailored to specific tasks especially for cases whose representations are under-represented

Lifecycle analysis which includes training cost, deployment capital cost, and operational overheads

Extension in multimodal tasks and more long form generation where SLM-LLM trade-offs may differ

10 Conclusion

This work has shown how technology (small language models) can substitute large models usefully in certain enterprise applications while maintaining the same performance of larger models and with dramatically reduced energy usage. Empirical evaluation in application contexts from code generation to email classification and sentiment analysis demonstrates that retention of high accuracy can be achieved and large energy savings obtained with SLMs and represents a realistic step towards increased sustainability in organisations using AI.

The results suggest that an SLM-first strategy is able to deliver immediate environmental benefits without changing existing infrastructure. By giving priority to the task characteristics and implementing models that meet particular workloads, organisations can get the best balance between performance and sustainability.

Future research will need to delve into greater task applicability and scaling of further processes, and potential for greater optimisation of training regimes. In addition, on-site empirical energy measurements in a wide range of deployment scenarios will allow further refinement in the understanding of the sustainability of AI systems.

References

ArXCompass. Codeif: Benchmarking the instruction-following capabilities of large language models for code generation. *arXiv preprint arXiv:2502.19166*, 2024.

M. A. Ashraf et al. Energy-aware code generation with llms: Benchmarking small vs. large models. *arXiv preprint arXiv:2508.08332*, 2025. doi: 10.48550/arXiv.2508.08332.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. doi: 10.48550/arXiv.2108.07732.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. doi: 10.48550/arXiv.2107.03374.

Jared Fernandez, Clara Na, Vashisth Tiwari, Yonatan Bisk, Sasha Luccioni, and Emma Strubell. Energy considerations of large language model inference and efficiency optimizations. *arXiv preprint arXiv:2504.17674*, 2025. doi: 10.48550/arXiv.2504.17674.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*, 2021. doi: 10.48550/arXiv.2105.09938.

Ines Jegham, Krishna Teja Chitty-Venkata, Siddhisanket Raskar, Bharat Kale, Farah Ferdaus, Aditya Tanikanti, Ken Raffenetti, Valerie Taylor, Murali Emani, and Venkatram Vishwanath. How hungry is ai? benchmarking energy, water, and carbon footprint of llm inference. *arXiv preprint arXiv:2505.09598*, 2025. doi: 10.48550/arXiv.2505.09598.

Maxime Labonne and Robert Moran. An improved transformer-based model for detecting phishing, spam, and malicious emails. *arXiv preprint arXiv:2311.04913*, 2023. doi: 10.48550/arXiv.2311.04913.

Yu Zhang et al. Sentiment analysis with large language models: Comparative evaluation of gpt-4, bert, and xlnet on social media text. *arXiv preprint arXiv:2305.15005*, 2023. doi: 10.48550/arXiv.2305.15005.

A Detailed Experimental Setup

A.1 Quantization Configuration

In accordance to the BitsAndBytes methodology, 4-bit quantisation has been used with the following configuration:

```
quantization_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_compute_dtype=torch.float16,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type="nf4"
)
```

A.2 Inference Parameters

For code generation: temperature = 0.2, top-p = 0.95, max tokens = 512.

For classification: temperature = 0.1, top-p = 0.9, max tokens = 10.

Different decoding strategies were used according to greedy decoding. To evaluate code generation, the standard HumanEval evaluation harness with a 10-second per test execution time was used, while classification tasks used exact match labeling.

B Additional Results

B.1 Email Classification Breakdown

The accuracy obtained in different categories of emails is shown in Table 7.

The variation in accuracy across categories is a reflection of the fact that, the domains with more balanced semantic indicators (HR General, Recruiting) have a higher performance rate. Categories with fewer representative samples such as Training and Benefits are less accurate suggesting that accuracy could be improved through data enrichment or specialised training.

Table 7: Email categorization accuracy from our experiments

Category	Samples	Accuracy (%)
HR General	234	82.1
Benefits	187	68.4
Payroll	156	71.8
Recruiting	198	75.3
Training	164	62.2

Table 8: Per-task energy consumption from our experimental estimates

Task	Total Energy (kWh)	Mean per Query (Wh)	CO_2 Reduction (kg)
Code Generation	35.37	0.015–0.025	15.5
Sentiment Analysis	15.28	0.001–0.002	6.7
Email Classification	2.40	0.003–0.005	1.05

B.2 Per-Task Energy Consumption

Estimated energy consumed per task is described in Table 8.

With these results, SLM deployment has the potential to bring massive energy savings, especially in high frequency operations like code generation and sentiment analysis. The associated reduction of CO2 is further proof of the environmental benefits of energy efficient inference strategies.

B.3 Extended Error Analysis

The error types that occurred and their frequencies are categorised in Table 9.

Table 9: Error categorisation from our experiments

Error Type	Occurrences	Description
Syntactic Errors	15	Issues with code syntax, punctuation omissions
Logic Faults	30	Faults in logic causing flawed test cases
Classification Errors	25	Mislabeling of email and sentiment
Edge Cases	10	Unable to handle atypical inputs

Logical errors during code creation are the biggest challenge for SLMs – they are usually the result of requiring a complex reasoning or containing edge cases. Mislabeling or insufficient generalisation on classification problems is primarily the result of insufficient exposure to training. These insights inform future model refinement and fine-tuning procedures.