Is ChatGPT Generated Code Really Green? Evaluating AI-Generated Solutions for Energy Efficiency

Aman Swaraj, *Indian Institute of Technology Roorkee, Roorkee, Uttarakhand, 247667, India*

Sandeep Kumar, *Indian Institute of Technology Roorkee, Roorkee, Uttarakhand, 247667, India*

Abstract—This document presents the preliminary free-form comparative analysis that informed the early version of our study on the energy efficiency of AI-generated code. These exploratory results were generated prior to the structured evaluation introduced in the revised manuscript, where the full and final analysis is now reported. The material here is shared as an extended supplement to provide transparency and traceability of the evolution of our methodology and insights.

PATTERNS FROM COMPARATIVE ANALYSIS

After analyzing all the responses, we identified 40 cases where the human answers were more energy-efficient. While ChatGPT outperformed human answers on most occasions, the specific examples of human wins also comprised valuable insights. We discuss these findings and their implications in greater detail in the following sections. Additionally, a table depicting the theme-wise human-AI win distribution can be found in our repository.

Factors Affecting AI wins:

ChatGPT responses consistently outperformed human answers in the comparative analysis, especially in cases where the problem aligned well with standard structures or algorithm-based advantages. Some of these cases involved the use of standardized APIs, pipeline-based data handling, and model conversion utilities tasks. Below, we list some key patterns along with representative examples.

- **Familiar, standardized APIs:** We found that ChatGPT was quite effective when questions were related to widely adopted libraries and APIs such as `TFLiteConverter`, `tf.data`, or pruning utilities like `tfmot.sparsity.keras`. For instance, in the question *"How to set prunable layers for tfmot.sparsity.keras.prune_low_magnitude?"*, ChatGPT precisely recommended using the `prunable_layer_types` argument, while the human answer relied on exploratory code and iterative debugging.
- **Pipeline-based optimizations:** Another observation from our analysis was related to data ingestion tasks, such as in the SO post "*Make tensorflow dataset from huge number of images(*.jpg) and labels(*.mat)*", ChatGPT constructed efficient pipelines using parallel interleaving and caching which can improve throughput and reduced the idle time of CPU. In contrast, the human response relied on manual iteration or NumPy-based preprocessing outside the TensorFlow pipeline. Similarly, in another query,*"How to efficiently read specific lines of a CSV?"*, ChatGPT directly leveraged `TextLineDataset.skip()` and `take()` methods, generating a compact and optimized pipeline. In contrast, the human answer read the file manually using Python's native file I/O and conditionals, which, although more flexible, takes more time and is also memory-intensive.
- **Avoiding redundant steps:** ChatGPT also omitted unnecessary file operations. For instance, in the question *"How to bulk write TFRecords?"*, it efficiently used batched writes with `tf.io.TFRecordWriter`, while the human answer included per-record writing and repeated encoding. In another quantization-related query, "*How to ensure TFLite Interpreter is only using int8 operations?*", ChatGPT included appropriate post-training quantization parameters and inspection code, whereas human answers added redundant profiling steps and custom inference loops.
- **Up-to-date practices and tooling:** In one question, when it was asked "*How to convert*

XXXX-XXX © IEEE
Digital Object Identifier 10.1109/XXX.0000.0000000

Lenet model h5 to .tflite?", ChatGPT recommended the latest Keras TFLite APIs and representative datasets, while the human response included deprecated flags.

Overall, these advantages of AI responses can be attributed to their large-scale training on online resources, exhaustive documentation, and awareness of common coding patterns.

Theme-Wise Analysis of Human-Win Cases

While ChatGPT produced more energy-efficient solutions in most cases, we identified several instances where human-written answers performed better. In this section, we group these human-win cases as per the earlier listed eight themes and discuss the key insights revealed.

- **Pre-trained Models:** In questions related to pre-training, human answers excelled in 8 out of 31 cases. In one of the questions, which stated "Why would validation loss be exceptionally high while fitting with EfficientNet?". The human answer correctly diagnosed two critical issues which ChatGPT missed, i.e., the misalignment between the loss function and the data generator's `class_mode` and not setting `training=True` for the frozen base model. These corrections not only avoided wasted GPU cycles but also prevented instability due to improper `BatchNorm` updates.
  Similarly, in the post "What is the right way to preprocess images in Keras while fine-tuning pre-trained models?", ChatGPT confused the necessity of manual BGR conversion and neglected to mention the model-specific `preprocess_input()` functions. However, the human answer clarified these details, referenced the correct documentation, and presented a time-efficient pipeline for fine-tuning.
- **Quantization:** In model quantization, human answers outperformed ChatGPT in 6 out of 26 cases. In one such question, "Convert Keras MobileNet model to TFLite with 8-bit quantization", the human answer emphasized the importance of quantization-aware training (QAT) to preserve accuracy. It explained how simulated quantization during training can enable the model to adjust to lower precision, which is vital for real-time inference on edge devices. In contrast, ChatGPT suggested post-training quantization, which is quicker to implement but more prone to performance degradation in complex models. While the AI-provided solution was technically correct, the human answer aligned more closely with long-term efficiency and quality-aware deployment, thereby reducing the need for re-training.
- **Distillation:** In the context of distillation-related queries, human answers proved more energy-efficient in 4 out of 10 cases. For example, in the question "Unable to create group (name already exists)", ChatGPT focused narrowly on the error message and did not even consider the angle of a distillation-based approach. However, the human answer proposed a custom implementation using an AttentionMaps Keras layer to extract and compare intermediate features that can help in improving knowledge transfer between models. Instead of focusing only on the file-saving error, the human answer reframed the problem to support more efficient model compression.
  A similar example of effective human guidance can also be seen in the question titled "Change custom loss parameter and NN parameter with respect to epoch".
- **Efficient Read-Write:** Human answers outperformed ChatGPT in 4 out of 20 questions concerning I/O tasks. In one of the questions, "How to unpack an integer bit-pattern into a tf.Tensor?", the human answer introduced an innovative vectorized bitmasking method to extract all bits at once across a tensor, thereby avoiding the slower loop-based unpacking. This approach is both memory and time efficient and can be crucial in handling large-scale inputs. The LLM answer, while correct and easier to follow, used an iterative method that may not scale well in computationally intensive scenarios.
  In another case, "TensorFlow takes too long to load data into a tf.Dataset," the human answer recommended practical adjustments to the input pipeline that aligned well with TensorFlow's performance behavior. In contrast, ChatGPT completely overlooked prefetch(), missing a crucial point for improving asynchronous data loading performance.
- **Memory Leaks:** In our case study, there were 5 out of the 52 questions related to memory leaks where human answers outperformed ChatGPT. For example, in one of the questions asking "What is the most efficient way to interchange the values of two variables in a Tensorflow graph?", the human answer used `read_value()` along with `tf.control_dependencies` to correctly swap variables without causing memory leaks.

This method prevents intermediate tensors from being unintentionally retained in the computation graph, which is an important consideration in GPU settings. In contrast, ChatGPT used a straightforward assignment approach that mistakenly wrote both variables with the same value and ignored TensorFlow's execution order semantics. This can result in incorrect behavior and potential resource leaks.

In another example concerning "Memory leak when running Python script from C++," the human answer offered a detailed explanation of Python's garbage collection behavior and also how internal references in TensorFlow can prevent across language boundaries. Rather than just addressing the symptom, the answer helped developers understand the root cause and design more leak-resilient pipelines.

- **Tensor Operations:** In questions related to Tensor operations, we found 4 cases where human answers edged over the AI counterparts. In the post titled "Loops without unrolling in TensorFlow," the human answer simplified the loop structure by skipping the creation of separate condition and body functions, and thus resulted in a more compact and memory-efficient implementation. In contrast, ChatGPT followed the standard textbook approach using full `tf.while_loop` technique. While technically correct, this method can introduce unnecessary memory overhead in simpler use cases.
- **Pruning:** In pruning, a total of 6 out of the 16 human answers provided more effective and resource-specific solutions than the corresponding AI responses. For example, in the question "Prune computation graph in TensorFlow", the human answer recommended the standard energy-efficient method of using `strip_unused_nodes` from TensorFlow's graph transformation tools. This approach directly aligns with energy-saving principles as it removes all computation paths that may not be essential for producing the desired output. The ChatGPT response also mentioned working with GraphDef; however, it lacked the actionable insight of the `strip_unused_nodes` recommendation.

  In another question related to "Keras + TensorFlow Model Optimization. . . ", the human answer avoided the need for post hoc pruning by applying pruning correctly during the model construction phase itself. This approach reduced both code complexity and the risk of compatibility issues with newer APIs.
- **Checkpointing:** Finally, in the questions related to checkpoints, we found 3 out of 13 questions where human responses were better. For instance, in the question "Difference between Keras model.save() and model.save_weights()?", the LLM response gave proper information, but it also included subtle inaccuracies that could lead to energy-inefficient workflows. For example, it suggested that saving the model architecture and weights separately using `model.to_json()` and `model.save_weights()` is equivalent to `model.save()`. However, as the human answer correctly pointed out, this approach leaves out important components such as the optimizer state and compile settings, and this can make it difficult to resume training without manually reconfiguring the model. As a result, it can lead to unnecessary retraining or debugging.

Overall, in these examples, we saw how human answers performed better in specific situations involving non-standard configurations, undocumented behavior, or platform-specific limitations.

While we acknowledge that some of these human-win cases may be influenced by chance or specific contextual factors, nevertheless, we argue that these cases still offer valuable indicative remarks pointing to areas where AI-generated assistance can be improved and where human-AI collaboration may be especially effective. Based on these findings, in the next section, we list broader takeaways for tool designers and software practitioners.

Authors -

Aman Swaraj is a PhD student at the Indian Institute of Technology Roorkee, Roorkee-247667, India. His research interests include information retrieval utilizing natural language processing and computer vision techniques, as well as human-centered studies in computing. Contact him at aman_s@cs.iitr.ac.in.

Sandeep Kumar is a professor in the Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, Roorkee 247667, India. His research interests include web services, software engineering, and artificial intelligence. He is a Senior Member at IEEE and ACM. Contact him at sandeep.garg@cs.iitr.ac.in.