## Prediction Model Building Process

### Parameters planning

The model needs specific, carefully chosen parameters to effectively capture the workload that an LLM (or other technologies) must manage for query processing. This workload is then expressed in terms of energy consumption, water usage, and carbon footprint.

The most important factors that directly relate to this are,

* Query length
* Query type (code gen, image analysis, image gen, doc analysis, etc..)
* Model name
* Region of server that responds
* No. of API calls (Gemini does this)
* Part of the day at the data center (morning/afternoon/night)
* Season present at data center area

Output of the model:

* Power consumption (in kWh)
* Water consumption (in L)
* CO2 emission (in Kg CO2)

These outputs should then be mapped to real-world examples for easy understanding.

Query Type prediction model:

The query type of each query passed will be determined by a classification model that is trained on synthetic data that was generated from a real-world sample population collected from a survey. It uses Vertex AI’s AutoML model and gretle tool.

### Data Collection for Each Parameter

To ensure a comprehensive dataset for our model, we employ a combination of simulation, publicly available data, and reasonable assumptions.

* **Query Length and Type**: Simulated using a range of predefined structures and randomly generated query formats.
* **Model Names**: Selected from a curated list of commonly used AI models.
* **Server Regions / Data Centers**: Extracted from publicly available datasets, such as those on Kaggle, or sourced from cloud provider documentation.

The challenge is in simulating the corresponding outputs for each input.

No information is available for predicting the output parameters. Cloud service providers did not provide open data on these metrics. So, we are making some assumptions and generalizations for predicting the outputs.

**DISCLAIMER:**

The predicted sustainability metrics in this model—**power consumption, water footprint, and CO₂ emissions**—are **approximations** based on simulated data and estimations. These values **do not represent real-time or provider-specific measurements** but serve as indicative benchmarks for understanding AI’s environmental impact.

### Methods used to simulate output

Using Publicly available data,

* Google Cloud carbon footprint reports
* Microsoft’s Sustainability Calculator
* Top cloud service provider’s energy and sustainability reports
* Research papers on LLM energy usage
* Stanford AI index report
* Open-source datasets available

### Research Papers to analyze

Power consumption

<https://ieeexplore.ieee.org/abstract/document/10363447>

<https://arxiv.org/abs/2205.09646>

<https://arxiv.org/abs/2407.04014>

<https://www.sciencedirect.com/science/article/pii/S2210537923000124>

<https://dl.acm.org/doi/abs/10.1145/3630106.3658542>

<https://arxiv.org/abs/2109.05472>

Carbon Footprint analysis

<https://dl.acm.org/doi/abs/10.1145/3604930.3605705>

<https://dl.acm.org/doi/fullHtml/10.1145/3603746>

<https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5036344>

<https://arxiv.org/pdf/2309.14393>

Water consumption

<https://puiij.com/index.php/research/article/view/39>

<https://sustainability.biruni.edu.tr/sites/default/files/2024-05/Gupta%2C%20et%20al._AIs%20excessive%20water%20consumption.pdf>

<https://dl.acm.org/doi/abs/10.1145/3578337.3605121>

### Inference from research papers

For LLaMA model, which is similar to chatGPT3

* 65B (65 billion parameters, 80 layers, 8192 dimensions)
* words/sec or response/second **inversely proportional** to model parameter size
* Advanced hardware (A100 GPUs) **proportional** to energy usage but less response time
* On bare minimum hardware settings,
  + 65B: 460 W/sec (approx.)
  + 13B: 380 W/sec (approx.)
  + 7B: 200W/sec (approx.)
* On best case,
  + Approximately 1000W/sec

Model training of chatGPT - 1300MWh (approx.)

Datacenter PUE:

PUE = (FE+ITE)/ITE, FE - facility energy; ITE - information tech. Energy

PUE high during afternoons and summers

* PUE = 1.6 (afternoons) ; PUE = 1.2 (night)
* PUE = 1.3 to 1.4 during summer
* Major datacenter like google and other, PUE = 1.1

GFLOPs are more useful in compute than no. of parameters

Correlation between GFLOPs and # of parameters

* Transformers - 0.994
* CNNs - 0.772

Top 1-accuracy models have GFLOPs around 50 to 100(0.3 to 1 Joule)

Very high accuracy (85% - 90%) - around 1000 GFLOPs(30 Joules)  
  
Tokens play an important role in energy consumption

* Input tokens (8 tokens -> 80J/token ; 2^11 tokens -> 20J/token)
* Output tokens (8 tokens -> 10J/token ; 2^11 tokens -> 700J/token)

Output tokens play a crucial role in energy consumption.

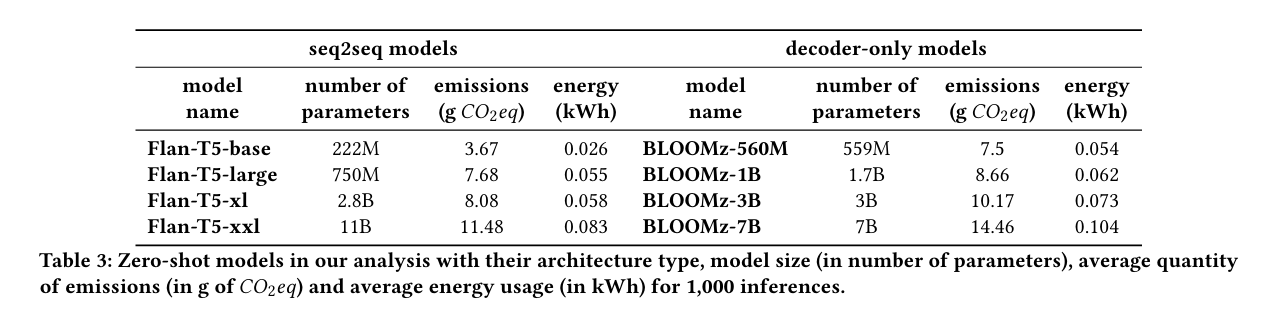
The number of parameters is usually reported, but it is not directly proportional to compute. For instance, in [CNNs](https://www.sciencedirect.com/topics/engineering/convolutional-neural-network), convolution operations dominate the computation: if 𝑑, 𝑤 and 𝑟 represent the network’s depth, width and input resolution, the FLOPs grow following the relation:

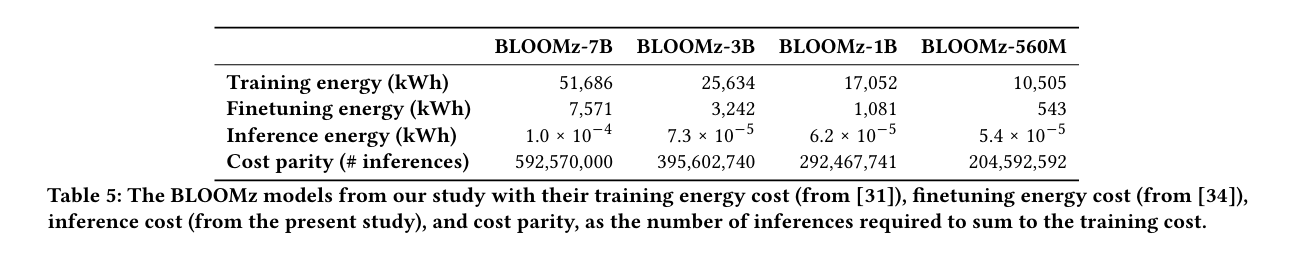
FLOPs ∝ 𝑑+𝑤^2+𝑟^2

inference energy (kWh)

| task | Mean | Std. Deviation |
| --- | --- | --- |
| text classification | 0.002 | 0.001 |
| extractive QA | 0.003 | 0.001 |
| masked language modeling | 0.003 | 0.001 |
| token classification | 0.004 | 0.002 |
| image classification | 0.007 | 0.001 |
| object detection | 0.038 | 0.02 |
| text generation | 0.047 | 0.03 |
| summarization | 0.049 | 0.01 |
| image captioning | 0.063 | 0.02 |
| image generation | 2.907 | 3.31 |

Table: Mean and standard deviation of energy per 1,000 queries for the ten tasks examined in our analysis.

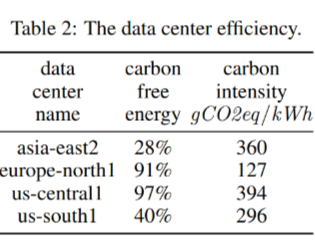


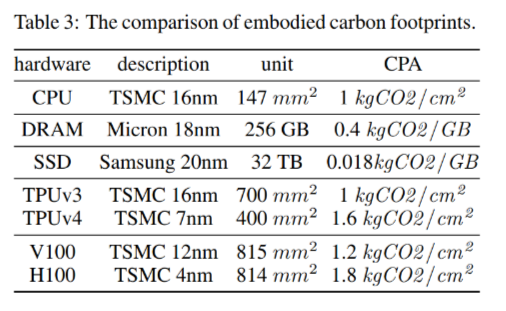
BLOOMz-7B BLOOMz-3B BLOOMz-1B BLOOMz-560M Training energy (kWh) Finetuning energy (kWh) Inference energy (kWh) Cost parity (# inferences) 51,686 7,571 1.0 × 10−4 592,570,000 25,634 3,242 7.3 × 10−5 395,602,740 17,052 1,081 6.2 × 10−5 292,467,741 10,505 543 5.4 × 10−5 204,592,592  


Datacenter carbon intensity (gCO2e/kWh)

### Government & Tech Reports

* **Google Cloud Sustainability Reports** (Google’s Environmental Report)
* **Microsoft Sustainability Calculator** ([Microsoft Cloud CO₂ Emissions](https://www.microsoft.com/en-us/sustainability/emissions-impact-dashboard))
* **AWS Carbon Footprint Report** (AWS Sustainability)
* **International Energy Agency (IEA) Reports** (IEA AI & Energy)





## Query classification Model

### **Project Direction**

The primary objective of the project evolved from developing a model to predict sustainability metrics to building a robust query classification model. During the course of development, we identified that sustainability metrics can be accurately computed using predefined formulas and findings from established research literature. However, classifying user-provided queries into meaningful categories—such as text generation, code generation, summarization, etc.—posed a more complex challenge that could not be addressed using rule-based methods. This necessitated the development of an AI-based solution specifically for query classification.

### **Data Collection**

As there were no publicly available datasets containing user queries labeled with corresponding categories, we undertook the task of curating our own dataset. Initial data was collected via Google Forms, where we gathered real-world user queries along with their respective labels. To augment this dataset, we employed generative techniques to synthesize additional query samples. Furthermore, we utilized **Gretel.ai**, a synthetic data generation platform, to expand our dataset at scale while preserving label relevance and data diversity.

### **Model Training and Deployment**

The compiled dataset was uploaded to **Vertex AI**, where we trained a custom **AutoML classification model**. The AutoML approach allowed us to leverage Google's advanced machine learning infrastructure without requiring deep model customization or manual hyperparameter tuning.

Upon successful training, the model was deployed to a **Vertex AI endpoint**. This endpoint facilitates both real-time and batch predictions, enabling seamless integration into our application and allowing us to efficiently classify user queries in production environments.