LARGE LANGUAGE MODELS FOR HUMAN-MACHINE COLLABORATIVE PARTICLE ACCELERATOR TUNING THROUGH NATURAL LANGUAGE

A PREPRINT

Jan Kaiser[*1], **Annika Eichler**[†1,2], **and Anne Lauscher**[‡3]
[1]Deutsches Elektronen-Synchrotron DESY, Germany
[2]Hamburg University of Technology, 21073 Hamburg, Germany
[2]Universität Hamburg, Germany

14 May 2024

ABSTRACT

Autonomous tuning of particle accelerators is an active and challenging field of research with the goal of enabling novel accelerator technologies cutting-edge high-impact applications, such as physics discovery, cancer research and material sciences. A key challenge with autonomous accelerator tuning remains that the most capable algorithms require an expert in optimisation, machine learning or a similar field to implement the algorithm for every new tuning task. In this work, we propose the use of large language models (LLMs) to tune particle accelerators. We demonstrate on a proof-of-principle example the ability of LLMs to successfully and autonomously tune a particle accelerator subsystem based on nothing more than a natural language prompt from the operator, and compare the performance of our LLM-based solution to state-of-the-art optimisation algorithms, such as Bayesian optimisation (BO) and reinforcement learning-trained optimisation (RLO). In doing so, we also show how LLMs can perform numerical optimisation of a highly non-linear real-world objective function. Ultimately, this work represents yet another complex task that LLMs are capable of solving and promises to help accelerate the deployment of autonomous tuning algorithms to the day-to-day operations of particle accelerators.

Keywords Large language models · Autonomous particle accelerators · Multi-objective optimisation

1 Introduction

Particle accelerators are sophisticated machines designed to accelerate subatomic particles, such as electrons and protons, to extremely high speeds, often close to the speed of light. These devices play a crucial role in a variety of applications, ranging from fundamental research in physics to practical uses in medicine, such as cancer therapy, and material science. As the demands from these diverse applications grow, there is an increasing need for advanced tuning and control methods to manage the complex dynamics of particle acceleration. Despite this, as a result of its complexity, the tuning of particle accelerators is to this day often done manually by experienced human operators. In this context, the emergence of autonomous tuning methods represents a significant advancement. By leveraging methods from the fields of numerical optimisation and machine learning [Emery et al., 2003, Roussel et al., 2023a, Kaiser et al., 2022], autonomous systems promise to speed up accelerator tuning procedures, reducing costs and minimising downtime, while also enabling novel operating modes for state-of-the art measurements. Moreover, such methods enable a paradigm shift from actuator-driven accelerator operation, where human operators control actuator settings to achieve good measurement conditions, to specification-driven operation, where human operators determine the best conditions for

[*]jan.kaiser@desy.de
[†]annika.eichler@desy.de
[‡]anne.lauscher@uni-hamburg.de

experiments and autonomous agents ensure that these conditions are achieved. As such, autonomous particle accelerator tuning methods promise to not only improve the performance of accelerators on existing applications but also open up new possibilities in scientific research and industrial applications, marking a transformative step in the field of particle acceleration.

However, implementing state-of-the-art accelerator tuning methods on new tuning tasks requires experts in two separate domains – accelerator physics and optimisation – as well as significant engineering effort to solve problems ranging from algorithm selection to objective function formulation. These challenges have so far slowed the adoption of advanced autonomous tuning algorithms to day-to-day accelerator operations.

In recent developments, LLMs, such as *GPT 4* [OpenAI et al., 2023] and *Llama 2* [Touvron et al., 2023], have been demonstrated to be capable of solving complex tasks when prompted through natural language [Brown et al., 2020, OpenAI et al., 2023, Oulianov et al., 2024]. The question arises whether LLMs can directly perform particle accelerator tuning, when prompted by an accelerator expert describing the tuning goal. If capable, this would provide a more natural way of controlling autonomous tuning solutions through natural language, potentially enabling a more straightforward deployment of autonomous particle accelerator tuning solutions, and removing the requirement for optimisation algorithm-specific expertise. Moreover, the ability of LLMs to explain their reasoning [Wei et al., 2023] could provide valuable insights into the complex dynamics of particle accelerators, potentially aiding human operators in understanding the tuning process. Lastly, the successful application of LLMs to particle accelerator tuning would also demonstrate the ability of LLMs to solve (multi-objective) numerical optimisation problems, possibly opening up new avenues for the application of LLMs to optimisation tasks beyond particle accelerators.

In this work, we introduce a novel approach to using LLMs for autonomous tuning of a particle accelerator. We answer whether current state-of-the-art LLMs are in fact capable of solving particle accelerator tuning tasks and evaluate our LLM-based approach against the current state of the art in accelerator tuning using RLO and BO.

To this end, we review related work in Section 2, before introducing our approach for autonomous tuning of a particle accelerator using LLMs and our prompt design in Section 3. In Section 4, we evaluate the developed solution, comparing 14 state-of-the-art LLM models against each other; against 3 state-of-the-art accelerator tuning solutions, RLO, BO and extremum seeking (ES); as well as against two baselines, random search and doing nothing. Our findings indicate that LLMs are capable of tuning particle accelerators, but do not yet achieve performance competitive with the state of the art. We conclude this paper and discuss opportunities for future applications of LLMs in the operation of particle accelerator facilities in Section 5.

2 Related Work

Initial efforts towards autonomous accelerator tuning have investigated numerical methods such as Nelder-Mead simplex [Emery et al., 2003, Shang and Borland, 2005, Huang, 2018], robust conjugate direction search (RCDS) [Huang et al., 2013, Olsson et al., 2018, Zhang et al., 2022a], extremum seeking (ES) [Scheinker et al., 2022] and genetic algorithms [Bergan et al., 2019]. These methods have since found adaptation in the day-to-day tuning of particle accelerator facilities [Tomin et al., 2016, Zhang, 2021, Zhang et al., 2022b]. More recently, advanced methods like Bayesian optimisation (BO) have found increased interest in the accelerator community [Roussel et al., 2023a] for their ability to utilise machine learning to learn a probabilistic surrogate model of the underlying objective function, enabling more sample-efficient tuning of high-dimensional and increasingly complex accelerator systems. Efforts are currently under way to lower the barrier of entry to these methods and increase their adoption in day-to-day accelerator operations [Roussel et al., 2023b]. Moreover, the accelerator community is looking increasingly to machine learning methods to aid with the challenges of accelerator tuning [Edelen et al., 2018]. In particular, reinforcement learning (RL) has found adoption to accelerator control tasks [Boltz et al., 2020, St. John et al., 2021]. RL has also been successfully applied to so-called reinforcement learning-trained optimisation (RLO), where neural network (NN) policies are trained through *optimiser learning* [Andrychowicz et al., 2016, Li and Malik, 2017a,b, Chen et al., 2022] to be capable of sample-efficient accelerator tuning [Kain et al., 2020, Pang et al., 2020, Kaiser et al., 2022, Velotti et al., 2023].

Most recently, large language models (LLMs) have had a highly visible impact on the field of artificial intelligence (AI) and machine learning (ML). Usually based on the transformer NN architecture, first introduced in Vaswani et al. [2017], these models are trained to perform text completion, such that they develop text understanding and text generation capabilities, which can be exploited to create chatbots. As such, state-of-the-art LLMs like *GPT 4* [OpenAI et al., 2023] have been demonstrated to have impressive capabilities, such as text summarisation, but also the ability to solve more complex tasks like coding and general problem solving. The field of LLMs is moving fast and seeing significant investments. Despite their high training cost, many of these models have been released in a short time frame, both commercial and closed in nature, such as *GPT 4* [OpenAI et al., 2023], *Gemini* [Gemini Team et al., 2023] and *Claude* [Anthropic, 2023], but also numerous open-source (or more specifically open-weights) models, such as *Llama*

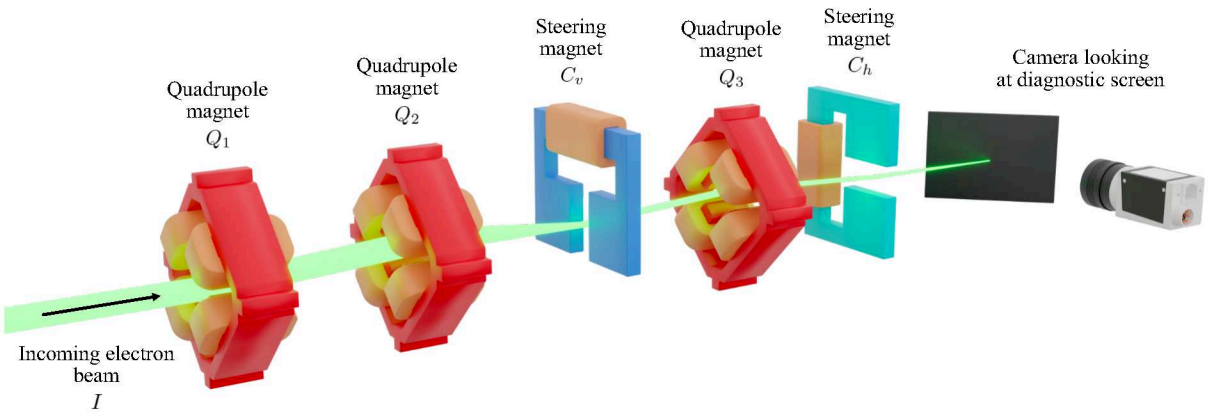

Figure 1: Schematic of the EA section of the ARES linear particle accelerator. Quadrupole magnets are shown in red; the vertical and horizontal dipole are shown in blue and turquoise, respectively. The electron beam is shown as a green envelop passing through the magnets and onto the screen at the end of the experimental area.

(2) [Touvron et al., 2023], *Orca* (2) [Mukherjee et al., 2023, Mitra et al., 2023], *Starling-LM* [Zhu et al., 2023] and *Mistral / Mixtral* [Jiang et al., 2023, 2024]. Most of these are released in varying sizes with varying trade-offs between capabilities and computational efficiency.

The application of LLMs to optimisation is less prominent in recent research than other applications. Naturally fitting the natural language processing (NLP) origins of LLMs, they have successfully been applied to optimising prompts to LLMs chatbots [Yang et al., 2023]. In further work, LLMs have been used to find more effective algorithms than the state of the art to solve complex problems [Romera-Paredes et al., 2024]. Most similar to our work, the ability of LLMs to solve numerical optimisation has been demonstrated on the simple task of linear regression in Yang et al. [2023]. A benchmark evaluating the performance of different LLMs on a game playing task like those typically solved by training NN policies through RL is presented in Oulianov et al. [2024].

In the context of particle accelerators, there exist ambitions to harness the NLP abilities of LLMs for various purposes. In Sulc et al. [2023], the authors demonstrate how to fine-tune an open-source LLM to be a particle accelerator domain expert using open access scientific literature as training data, augmented by another LLM to generate question-answer pairs from research papers. The fine-tuned model, called *PACuna*, is shown to be more proficient in answering questions related to particle accelerators. In Mayet [2024a,b], the author demonstrates how off-the-shelf LLMs can be used as a general AI assistant for intelligent accelerator operations (GAIA), employing the *ReAct* [Yao et al., 2023] prompting scheme to enable the LLM to intelligently trigger accelerator operation routines, automatically contact experts when needed, research questions in the facility's logbook, provide the correct control system addresses for actuators and sensors of the accelerator, and write weekly shift reports.

3 Tuning Particle Accelerators Through Natural Language

For the purpose of this work, we consider a specific particle accelerator tuning task, namely the transverse beam parameter tuning in the Experimental Area (EA) section of the accelerator research experiment at SINBAD (ARES) linear particle accelerator [Panofski et al., 2021, Burkart et al., 2022] at DESY in Hamburg, Germany. This task has been chosen as it is a well-defined and well-understood task in the accelerator community, and has been extensively studied in the context of autonomous accelerator tuning [Kaiser et al., 2022, 2023, Kaiser and Xu, 2023, Xu et al., 2023]. At the same time, the task is complex enough to be difficult to solve manually and can provide a meaningful benchmark for the capabilities of LLMs in accelerator tuning, yet simple enough such that solutions can still be easily understood and evaluated. Solving it would provide a valuable streamlining of accelerator operations because similar transverse tuning tasks can be found at most accelerator facilities and have to be regularly performed during everyday operations.

The EA section is primarily made up of five magnets as shown in Fig. 1. Three of these magnets are quadrupole magnets Q_1, Q_2 and Q_3, which are used to focus the beam, and two are dipole magnets C_v and C_h, which are used to bend the beam, one in the vertical and one in the horizontal plane. In this work, we control the focusing strength k of the quadrupole magnets in m^{-2} and the angle α by which the dipole magnets deflect particles in mrad. Note that turning up the strength of a quadrupole magnet will focus the beam in the horizontal plane and defocus it in the vertical plane, while turning down the strength will have the opposite effect. Increasing the steering angle of the

vertical steering magnet will steer the beam upwards, while decreasing the angle will steer the beam downwards. The horizontal steering magnet works similarly, steering the beam to the right when the angle is increased and to the left when the angle is decreased. What is more, quadrupole magnets also have a dipole effect on the beam, if the beam passes through the off-centre, making any tuning task involving them more complex. The magnets are arranged in the order $(Q_1, Q_2, C_v, Q_3, C_h)$. At the end of the EA section, there is a diagnostic screen station. At the screen station, a screen made of a scintillating material is inserted into the beam pipe. When electrons pass through the screen, light is emitted, which is then captured by a camera and used to measure a transverse projection of the beam. Transverse beam parameters of beam position $\mu_{x,y}$ and beam size $\sigma_{x,y}$ can then be computed from the screen image by fitting a 2D Gaussian distribution. The goal of the tuning task is to find a set of magnet settings $(k_{Q_1}, k_{Q_2}, \alpha_{C_v}, k_{Q_3}, \alpha_{C_h})$ that minimise the difference between the measured beam parameters $(\mu_x, \sigma_y, \mu_y, \sigma_y)$ and some target beam parameters $\left(\mu'_x, \sigma'_y, \mu'_y, \sigma'_y\right)$ set by the human operator.

3.1 Optimisation Scheme

We consider an iterative optimisation scheme for accelerator tuning, where the state of the accelerator is observed and then the tuning algorithm chooses new actuator settings based on the current and all past states from the tuning run. This process is repeated either for a fixed number of iterations or until some termination criterion is met. For an LLM to act as the tuning or optimisation algorithm, a prompting scheme needs to be devised. In our approach we consider the use of a chatbot LLM, where the user can provide a question or command to the LLM and the LLM will respond with an answer. Our optimisation scheme using LLMs extends on the approach for linear regression presented in Yang et al. [2023] and is shown in Fig. 2. In the prompt to the LLM, the user provides a description of the optimisation task that the LLM should solve. This is followed by a list of input and output pairs from previous optimisation steps. After this list, the user asks for the next set of input parameters that help optimise the objective function and gives the LLM instructions on how these should be formatted such that the user can parse the output from text to numerical values. This prompt is then sent to the LLM, which responds with the next set of input parameters that should be used to optimise the objective function, and potentially also an explanation of why these parameters were chosen. The response should look similar to the one below:

````
```json
{
  "Q1":  -14.30,
  "Q2":  -9.70,
  "CV":  -2.55,
  "Q3":  -8.10,
  "CH":  -5.21
}
```
I suggest decreasing Q1 slightly to bring down the horizontal beam position, while keeping
the other quadrupole magnets at their previous values to maintain the vertical beam position
and focusing.  I also kept the steering magnet settings close to their last values for
smoothness.
````

The response is then parsed, and the input parameters are used to evaluate the objective function. The output of this evaluation is then added to the list along with its corresponding input parameters, and the process is repeated.

Prompt engineering is a crucial part of using LLMs and can significantly impact the performance of the model. Because of the variability in the performance of different prompts and the difficulty of finding the best prompt, we evaluate the ability of LLMs to solve the accelerator tuning task using four different prompts: *Tuning Prompt* (see Section 3.1.1), *Explained Prompt* (see Section 3.1.2), *Chain-of-Thought Prompt* (see Section 3.1.3) and *Optimisation Prompt* (see Section 3.1.4). All prompts follow the general prompting scheme described above, of task description, input-output pairs, request for next input parameters and instructions on how to format the output. The prompts used in this work differ mainly in the task description and the outputs of the previous optimisation steps.

3.1.1 Tuning Prompt

The *Tuning Prompt* is the most straightforward and intuitive prompt used in this work. It describes the task of tuning the transverse beam parameters in the EA section and the goal of achieving some target beam parameters on the diagnostic screen, such that the LLM is aware of the fact it is tuning a particle accelerator. The input-output pairs are the magnet settings and the corresponding measured beam parameters. This prompt assumes that the LLM has some understanding of particle accelerators and understands, for example, what a quadrupole magnet is and how it affects the beam. Below

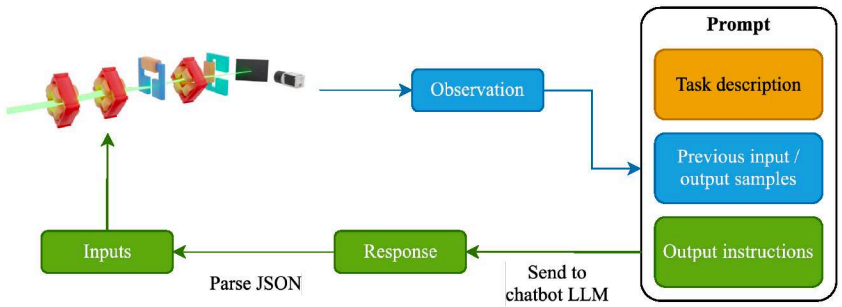

Figure 2: Flowchart of the optimisation scheme used to tune particle accelerators using LLMs. The prompt is made up for three components: Task description, list of previous input and output samples, and instructions what to output and how to format the output. The prompt is then sent to the LLM, which generates a response. The response is parsed into values that can be input into the tuning / optimisation task. A measurement or objective value from the task is then inserted into the previous samples along with its corresponding input and the loop is repeated.

is an example of the Tuning Prompt, where the task description is highlighted in orange, the input-output pairs in blue, and the request for the next input parameters and output instructions in green:

````
Human: Now you will help me optimise the horizontal and vertical position and size of an
electron beam on a diagnostic screen in a particle accelerator.

You are able to control five magnets in the beam line. The magnets are called Q1, Q2, CV, Q3
and CH.

Q1, Q2 and Q3 are quadrupole magnets. You are controlling their k1 strength in m^-2. Their
range is -30.0 to 30.0 m^-2.

CV is vertical steering magnet. You control its steering angle in mrad. Its range is -6.0 to
6.0 mrad.

CH is horizontal steering magnet. You control its steering angle in mrad. Its range is -6.0
to 6.0 mrad.

You are optimising four beam parameters: mu_x, sigma_x, mu_y, sigma_y. The beam parameters
are measured in millimetres (mm). The target beam parameters are:

Target beam parameters:
```json
{
  "mu_x": 1.20,
  "sigma_x": 0.11,
  "mu_y": 1.25,
  "sigma_y": 0.06
}
```

Below are previously measured pairs of magnet settings and the corresponding observed beam
parameters.

Magnet settings:
```json
{
  "Q1": 25.12,
  "Q2": 12.48,
  "CV": 0.84,
  "Q3": -8.25,
  "CH": 3.94
}
````
````

```
```

Beam parameters:
```json
{
  "mu_x": -1038.63,
  "sigma_x": 1893.75,
  "mu_y": -2353.77,
  "sigma_y": 2226.94
}
```

Give me new magnet settings that are different from all pairs above. The magnet settings you
should propose should lead to beam parameters closer the target or, if you do not have
enough information yet, maximise information gain for finding new beam parameters. Do not
set any magnet setting to zero. Smooth changes relative to the last magnet settings are
preferred.

The output should be a markdown code snippet formatted in the following schema, including
the leading and trailing "```json" and "```":

```
{
  "Q1": float // k1 strength of the first quadrupole magnet
  "Q2": float // k1 strength of the second quadrupole magnet
  "CV": float // Deflection angle of the vertical steering magnet
  "Q3": float // k1 strength of the third quadrupole magnet
  "CH": float // Deflection angle of the horizontal steering magnet
}
```

Do not add comments to the output JSON.
```

Note that the choice was made to provide previously observed magnet settings and beam parameters formatted as a markdown JSON snippet. We found that if these are provided as a simple textual list of property names and their values, the LLMs would often output the next magnet settings in the same format instead of the requested JSON format. By providing the examples in the same format as we desire for the output, the parsing reliability of the LLM is increased significantly.

### 3.1.2 Explained Prompt

The *Explained Prompt* is mostly the same as the Tuning Prompt, but includes additional explanations of how each of the magnets affects the beam. This is done because accelerator physics is a complex and niche field, which is unlikely to have been widely covered in the training data of a general-purpose LLMs. The explanations are generally kept on a high level, similar to how one might explain the task to a new accelerator operator in order to give them an intuition of how the magnets affect the beam on the diagnostic screen. Below is an example of the Explained Prompt with the explanations added over the Tuning Prompt highlighted in violet:

```
Human: Now you will help me optimise the horizontal and vertical position and size of an
electron beam on a diagnostic screen in a particle accelerator.

You are able to control five magnets in the beam line. The magnets are called Q1, Q2, CV,
Q3 and CH.

Q1, Q2 and Q3 are quadrupole magnets. When their k1 strength is increased, the beam becomes
more focused in the horizontal plane and more defocused in the vertical plane. When their k1
strength is decreased, the beam becomes more focused in the vertical plane and more
defocused in the horizontal plane. When their k1 strength is zero, the beam is not focused
in either plane. Quadrupole magnets might also steer the beam in the horizontal or vertical
plane depending on their k0 strength, when the beam does not travel through the centre of
the magnet. The range of the k1 strength is -30.0 to 30.0 mˆ-2.
```
```

````
CV is vertical steering magnet.  When its deflection angle is increased, the beam is steered
upwards. When its deflection angle is decreased, the beam is steered downwards. The range of
the deflection angle is -6.0 to 6.0 mrad.

CH is horizontal steering magnet.  When its deflection angle is increased, the beam is
steered to the right. When its deflection angle is decreased, the beam is steered to the
left. The range of the deflection angle is -6.0 to 6.0 mrad.

You are optimising four beam parameters:  mu_x, sigma_x, mu_y, sigma_y.  The beam parameters
are measured in millimetres (mm).  The target beam parameters are:

Target beam parameters:
```json
{
  "mu_x":  1.20,
  "sigma_x":  0.11,
  "mu_y":  1.25,
  "sigma_y":  0.06
}
```

Below are previously measured pairs of magnet settings and the corresponding observed beam
parameters.

Magnet settings:
```json
{
  "Q1":  25.12,
  "Q2":  12.48,
  "CV":  0.84,
  "Q3":  -8.25,
  "CH":  3.94
}
```
Beam parameters:
```json
{
  "mu_x":  -1038.63,
  "sigma_x":  1893.75,
  "mu_y":  -2353.77,
  "sigma_y":  2226.94
}
```

Give me new magnet settings that are different from all pairs above.  The magnet settings
you should propose should lead to beam parameters closer the target or, if you do not have
enough information yet, maximise information gain for finding new beam parameters.  Do not
set any magnet setting to zero.  Smooth changes relative to the last magnet settings are
preferred.

The output should be a markdown code snippet formatted in the following schema, including
the leading and trailing "```json" and "```":

```json
{
  "Q1":  float // k1 strength of the first quadrupole magnet
  "Q2":  float // k1 strength of the second quadrupole magnet
  "CV":  float // Deflection angle of the vertical steering magnet
  "Q3":  float // k1 strength of the third quadrupole magnet
  "CH":  float // Deflection angle of the horizontal steering magnet
}
```
````

```
Do not add comments to the output JSON.
```

3.1.3 Chain-of-Thought Prompt

Chain-of-Thought (CoT) prompting [Wei et al., 2023] is a technique where the user asks the LLM to explain its reasoning before it gives its answer. This was found to generally improve the quality of the answers given by LLMs, especially in the case of logical reasoning tasks. Note that it is important to have the explanation before the answer, as otherwise the model will phrase the explanation in support of the already given and potentially incorrect answer, thereby negating the benefit of chain-of-thought prompting. In the *Chain-of-Thought Prompt*, we add a request to the prompt whereby the users asks the LLM to explain its reasoning before it gives the next set of input parameters. Otherwise, the Chain-of-Thought Prompt is the same as the Explained Prompt. Below is an example of the Chain-of-Thought Prompt, where the request for chain-of-thought reasoning is highlighted in violet:

````
Human:  Now you will help me optimise the horizontal and vertical position and size of an
electron beam on a diagnostic screen in a particle accelerator.

You are able to control five magnets in the beam line.  The magnets are called Q1, Q2, CV,
Q3 and CH.

Q1, Q2 and Q3 are quadrupole magnets.  When their k1 strength is increased, the beam becomes
more focused in the horizontal plane and more defocused in the vertical plane.  When their
k1 strength is decreased, the beam becomes more focused in the vertical plane and more
defocused in the horizontal plane.  When their k1 strength is zero, the beam is not focused
in either plane.  Quadrupole magnets might also steer the beam in the horizontal or vertical
plane depending on their k0 strength, when the beam does not travel through the centre of
the magnet.  The range of the k1 strength is -30.0 to 30.0 mˆ-2.

CV is vertical steering magnet.  When its deflection angle is increased, the beam is steered
upwards.  When its deflection angle is decreased, the beam is steered downwards.  The range
of the deflection angle is -6.0 to 6.0 mrad.

CH is horizontal steering magnet.  When its deflection angle is increased, the beam is
steered to the right.  When its deflection angle is decreased, the beam is steered to the
left.  The range of the deflection angle is -6.0 to 6.0 mrad.

You are optimising four beam parameters:  mu_x, sigma_x, mu_y, sigma_y.  The beam parameters
are measured in millimetres (mm).  The target beam parameters are:

Target beam parameters:
```json
{
  "mu_x":  1.20,
  "sigma_x":  0.11,
  "mu_y":  1.25,
  "sigma_y":  0.06
}
```

Below are previously measured pairs of magnet settings and the corresponding observed beam
parameters.

Magnet settings:
```json
{
  "Q1":  25.12,
  "Q2":  12.48,
  "CV":  0.84,
  "Q3":  -8.25,
  "CH":  3.94
}
```
````

````
Beam parameters:
```json
{
  "mu_x":  -1038.63,
  "sigma_x":  1893.75,
  "mu_y":  -2353.77,
  "sigma_y":  2226.94
}
```

Give me new magnet settings that are different from all pairs above.  The magnet settings
you should propose should lead to beam parameters closer the target or, if you do not have
enough information yet, maximise information gain for finding new beam parameters.  Do not
set any magnet setting to zero.  Smooth changes relative to the last magnet settings are
preferred.

First, reason about how and why you would change the magnet settings in a certain direction.
Then give me the proposed magnet settings afterwards.

The output should be a markdown code snippet formatted in the following schema, including
the leading and trailing "```json" and "```":

```json
{
  "Q1":  float // k1 strength of the first quadrupole magnet
  "Q2":  float // k1 strength of the second quadrupole magnet
  "CV":  float // Deflection angle of the vertical steering magnet
  "Q3":  float // k1 strength of the third quadrupole magnet
  "CH":  float // Deflection angle of the horizontal steering magnet
}
```

Do not add comments to the output JSON.
````

3.1.4 Optimisation Prompt

The *Optimisation Prompt* phrases the task as a numerical optimisation problem instead of a particle accelerator tuning task. This means that the model is completely unaware that it is tuning a particle accelerator. Numerical optimisation tasks are more generic than particle accelerator tuning tasks and therefore expected to be more present in the training data used to train LLMs, meaning that models are likely to be more familiar with them. However, this also means that the model is given no information about the topology of the objective function, which makes the optimisation problem harder to solve. The objective function is therefore a black box to the model. The input-output pairs are the magnet settings and a corresponding single scalar objective value computed from the beam parameters as

$$\text{objective} = |\mu_x - \mu_x'| + |\mu_y - \mu_y'| + |\sigma_x - \sigma_x'| + |\sigma_y - \sigma_y'| \,. \tag{1}$$

Below is an example of the Optimisation Prompt, where the task description is highlighted in orange, the input-output pairs in blue, and the request for the next input parameters and output instructions in green:

````
Human: Now you will help me minimise a function with five input variables Q1, Q2, CV, Q3 and
CH. I have some (Q1, Q2, CV, Q3, CH) pairs and the corresponding function values at those
points. The samples are arranged in descending order based on their function values, where
lower values are better.

Inputs:
```json
{
  "Q1": -13.50,
  "Q2": -9.00,
  "CV": -3.00,
````
````

```
  "Q3": -9.00,
  "CH": -6.00
}
```
Objective value = 2.37

Inputs:
```json
{
  "Q1": -13.25,
  "Q2": -8.85,
  "CV": -2.80,
  "Q3": -8.90,
  "CH": -5.70
}
```
Objective value = 2.28

Give me a new sample (Q1, Q2, CV, Q3, CH) that is different from all pairs above, and has a function value lower than any of the above.

The output should be a markdown code snippet formatted in the following schema, including the leading and trailing "```json" and "```":

```json
{
  "Q1": float // First input
  "Q2": float // Second input
  "CV": float // Third input
  "Q3": float // Fourth input
  "CH": float // Fifth input
}
```

4 Evaluation

In order to evaluate whether LLMs using the prompting scheme described in Section 3 are capable of solving particle accelerator tuning tasks, we compare the performance of multiple state-of-the-art LLMs against each other and against other state-of-the-art accelerator tuning solutions using RLO and BO. In addition, we consider some further baselines for our comparison, specifically ES, random search and doing nothing. The evaluation setup is introduced in Section 4.1, followed by the evaluation results in Section 4.2.

4.1 Method

We evaluate each of the models and prompts on three different instances of the EA transverse beam parameter tuning task described in Section 3. We call these instances *trials*. Trials differ in the target beam parameters set by the human operator, the transverse misalignments of the quadrupole magnets and the diagnostic screen, the properties of the beam entering the EA section from upstream, and the initial magnet settings before the respective tuning algorithm has taken any action. For each trial, we run each model and prompt three times with different random seeds to account for the stochasticity of the LLMs and some of the baseline algorithms.

Performance is evaluated in terms of the mean absolute error (MAE) between the measured beam parameters and the target beam parameters after 50 iterations. This tests the ability of the models to find a good set of magnet settings. We further consider the normalised MAE improvement from the initial magnet settings to the final magnet settings found by the model, which tests the ability of the models to improve the beam parameters from the initial settings. Normalisation by dividing the MAE improvement by the MAE with the initial magnet settings makes this metric less sensitive to the inherent variability and difficulty of different trials. Finally, we consider the normalised MAE over all interactions, which tests the ability of the models to find a good set of magnet settings quickly. Here, too, the impact of trial-to-trial variations is reduced by dividing by the accumulated MAE of keeping the magnet settings the same as the initial settings for 50 iterations. For all LLMs, we further consider the number of consecutive steps for which they are

able to generate a parsable JSON output, which tests the tests the models' reliability in generating a valid output. LLMs are given a second chance in each sample, if they fail to generate a parsable JSON output on the first attempt.

The main goal of this work is not to determine whether LLMs are capable of outperforming the current state of the art in accelerator tuning algorithms. In fact, we expect that the current state of the art in accelerator tuning algorithms, such as RLO and BO, clearly outperform LLMs. Instead, we hope to determine whether LLMs are capable of solving accelerator tuning (and by extension other complex optimisation tasks) at all, and to what extent they can do so. We therefore also introduce three measures of "success", where we consider a tuning run successful, if the final beam difference is at least is $40\,\mu\mathrm{m}$ improved over the initial beam difference before any tuning has taken place, with $40\,\mu\mathrm{m}$ being twice the real-world measurement accuracy for beam parameters on the diagnostic screen. This means that runs are only considered successful, if a clearly measurable improvement of the beam parameters has been achieved. A tuning algorithm is considered "outright successful", if it is able to achieve the success criteria in all 9 evaluation runs. We consider a tuning algorithm as "partially successful" if it is able to achieve the success criterion in at least 6 of the 9 evaluation runs. Partial success suggests that, while not perfectly reliable, successful runs probably not coincidental. We further know that some trials can be harder to solve than others. As a third and weakest success criterion, we therefore consider a tuning algorithm as "single trial successful" if it is able to achieve the success criterion in on all three runs of a single trial, suggesting that, while some trials may have been too difficult to solve, the model was able to reliably solve this one trial.

For this work we evaluate a total of 14 different LLMs. These are *Gemma 2B* and *Gemma 7B* [Gemma Team et al., 2024] (version 1.0); *GPT 3.5 Turbo* (checkpoint 0125) [OpenAI, 2023], *GPT 4* [OpenAI et al., 2023] (checkpoint 0613) and *GPT 4 Turbo* (preview checkpoint 0125) [OpenAI, 2023]; *Llama 2 7B*, *Llama 2 13B* and *Llama 2 70B* [Touvron et al., 2023], as well as the fine-tuned variants of Llama 2: *Orca 2 7B* and *Orca 2 13B* [Mukherjee et al., 2023, Mitra et al., 2023], and *Vicuna 7B 16K* [Zheng et al., 2023]; *Mistral 7B* (version 0.2) [Jiang et al., 2023] and *Mixtral 8x7B* [Jiang et al., 2024]; and *Starling-LM 7B* (beta) [Zhu et al., 2023]. The Explained Prompt and Optimisation Prompt are evaluated with all models, while the Tuning Prompt and Chain-of-Thought Prompt are evaluated only with Gemma 2B, GPT 4 Turbo and Mixtral 8x7B.

Prompt generation and response parsing are implemented using the *LangChain* [Chase, 2022] Python package, which provides a straightforward set of tools for constructing prompts, calling LLMs and parsing their responses. The open-weights LLMs used in this work are run using *Ollama* [Ollama Team, 2023], while the OpenAI models are run through the OpenAI API. All models are run using their default temperature value, with $T = 0.7$ for the OpenAI models and $T = 0.8$ for all other models. Orca 2 7B, Orca 2 13B and Vicuna 7B 16K are run with their default system prompts as listed in Appendix A. All other models are run without any system prompts as per their default configuration. A *Gymnasium* [Farama Foundation, 2022] environment of the EA transverse beam parameter tuning task using the *Cheetah* [Stein et al., 2022, Kaiser et al., 2024] beam dynamics simulator is used to evaluate the LLMs and baselines. The baselines BO, ES and random search are implemented following Kaiser et al. [2023]. The RLO and do nothing baselines are implemented according to Kaiser et al. [2022], using the trained policy model from that work.

4.2 Results

The results of the evaluation in terms of the three previously defined metrics are shown in Table 1. The number of successful runs and wholly successful trials for each model and prompt are shown in Fig. 3. Two example tuning runs by a well-performing and a poorly-performing model and prompt combination are shown in Fig. 4.

We find that the state-of-the-art tuning algorithms RLO and BO, as well as ES, all achieve the strictest success criterion of outright success in all 9 evaluation runs. Of the LLM prompt combinations evaluated, GPT 3.5 Turbo, GPT 4 and GPT 4 Turbo in combination with the Optimisation Prompt also achieve outright success in all 9 evaluation runs, with GPT 4 Turbo with the Optimisation Prompt also being the best-performing LLM prompt combination in all evaluated metrics. In addition, a further 10 LLM prompt combinations achieve partial success, with Llama 2 13B, Llama 2 70B and Orca 2 7B doing so with the Optimisation Prompt; Gemma 7B, Mixtral 8x7B and Starling LM 7B achieving partial success with the Explained Prompt; Gemma 2B and Mixtral 8x7B achieving partial success with the Tuning Prompt; and Gemma 2B and GPT 4 Turbo achieving partial success with the Chain-of-Thought Prompt. Overall, Mixtral 8x7B is the best performing model with the Explained Prompt, but is outperformed by Starling LM 7B on the Final Beam Difference metric. With the Tuning Prompt, Mixtral 8x7B performs best of the three evaluated models, while Gemma 2B is the best-performing model with the Chain-of-Thought Prompt. All models that achieve partial success also achieve single trial success in at least one trial, demonstrating that they are able to solve some trials reliably. A further 6 LLM prompt combinations achieve single trial success: Gemma 2B and GPT 4 with the Explained Prompt, Mixtral 8x7B with the Optimisation Prompt, and Mixtral 8x7B with the Chain-of-Thought Prompt. In total, of the 34 LLM prompt combinations tried, 18 achieve at least one success criterion. Of 14 LLMs evaluated, 10 achieve at least one success criterion with at least one prompt. This demonstrates that LLMs can be used to solve accelerator tuning tasks.

Table 1: Evaluation Results

	Final beam difference (μm)		Normalised beam improvement (%)		Normalised integrated MAE (%)		Number of successful steps	
	Explained	Optimisation	Explained	Optimisation	Explained	Optimisation	Explained	Optimisation
Gemma 2B	1665 ± 634	3180 ± 5187	11 ± 71	34 ± 171	115 ± 51	137 ± 88	23 ± 19	39 ± 14
Gemma 7B	1651 ± 764	8105 ± 12933	-16 ± 11	284 ± 428	85 ± 10	247 ± 142	9 ± 0	29 ± 11
GPT 3.5 Turbo	11593 ± 14850	1197 ± 771	397 ± 618	-36 ± 27	292 ± 245	78 ± 20	**50** ± 0	**50** ± 0
GPT 4	1849 ± 1445	1213 ± 860	11 ± 73	-40 ± 25	98 ± 60	73 ± 20	**50** ± 0	**50** ± 0
GPT 4 Turbo	2184 ± 1879	**962** ± 740	20 ± 89	**-50** ± 28	107 ± 76	**67** ± 21	**50** ± 0	**50** ± 0
Llama 2 7B	1432 ± 798	2085 ± 779	-12 ± 55	15 ± 27	94 ± 15	106 ± 15	8 ± 6	3 ± 4
Llama 2 13B	1936 ± 772	1507 ± 821	5 ± 26	-22 ± 23	101 ± 10	85 ± 21	0 ± 1	13 ± 20
Llama 2 70B	1947 ± 964	1539 ± 942	10 ± 42	-21 ± 27	107 ± 37	92 ± 16	**50** ± 0	**50** ± 0
Orca 2 7B	2149 ± 1222	1377 ± 855	17 ± 42	-23 ± 37	122 ± 65	93 ± 17	4 ± 3	4 ± 7
Orca 2 13B	1634 ± 875	3232 ± 3684	-13 ± 24	77 ± 170	88 ± 18	142 ± 92	1 ± 2	3 ± 2
Vicuna 7B 16K	4756 ± 5332	4331 ± 3829	184 ± 320	320 ± 580	189 ± 137	234 ± 236	34 ± 7	48 ± 7
Mistral 7B	2551 ± 1233	19653 ± 23427	48 ± 57	803 ± 869	121 ± 40	1574 ± 1513	**50** ± 0	30 ± 22
Mixtral 8x7B	1606 ± 1158	1901 ± 1192	**-24** ± 27	-14 ± 31	**76** ± 26	101 ± 26	**50** ± 0	45 ± 14
Starling LM 7B	**1401** ± 449	7659 ± 7249	1 ± 69	363 ± 521	98 ± 60	324 ± 252	36 ± 15	19 ± 16
	Tuning	CoT	Tuning	CoT	Tuning	CoT	Tuning	CoT
Gemma 2B	1452 ± 525	**955** ± 702	-14 ± 46	**-40** ± 49	97 ± 43	87 ± 60	10 ± 1	**50** ± 0
GPT 4 Turbo	2647 ± 1827	1337 ± 813	45 ± 81	-23 ± 45	119 ± 64	**70** ± 25	**50** ± 0	**50** ± 0
Mixtral 8x7B	**1321** ± 771	1775 ± 926	**-29** ± 23	-8 ± 17	**71** ± 22	95 ± 19	**50** ± 0	**50** ± 0
Baselines								
RLO	**16** ± 17		**-99** ± 1		**3** ± 1		-	
BO	100 ± 26		-93 ± 6		31 ± 23		-	
Extremum seeking	457 ± 267		-71 ± 19		35 ± 17		-	
Random search	7677 ± 3830		487 ± 588		647 ± 476		-	
Do nothing	1967 ± 903		0 ± 0		100 ± 0		-	

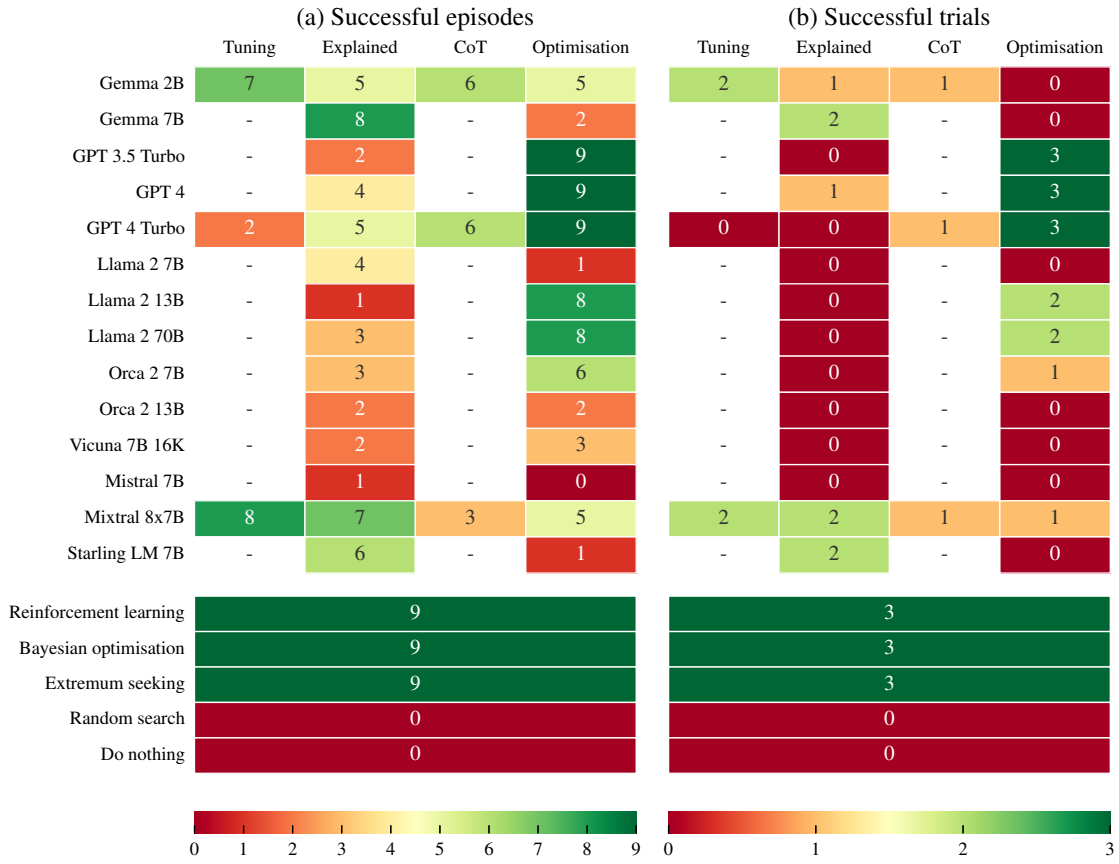

Figure 3: Number of successful runs for each model and prompt (a) and the number wholly successful trials, i.e. trials where all three runs were successful (b). We define as success an improvement of at least $40\,\mu\mathrm{m}$ on the beam differences when compared to the initial magnet settings.

However, these results also show that LLMs are not yet competitive with the state-of-the-art accelerator tuning algorithms. The best-performing LLM prompt combination, GPT 4 Turbo with the Optimisation Prompt, achieves an average normalised beam improvement of $-50\,\%$. This is a good result, but it is also a significantly worse result than the $-99\,\%$ and $-93\,\%$ achieved by RLO and BO, respectively. A similar trend can be observed in how fast algorithms are able to find a good set of magnet settings. GPT 4 Turbo with the Optimisation Prompt achieves an average normalised integrated MAE of $67\,\%$, which is an order of magnitude worse than the $3\,\%$ achieved by RLO. However, it is only about two times worse than BO and ES.

What is more, the results show that the performance of LLMs is highly dependent on the specific model and prompt used. While 18 of the 34 LLM prompt combinations tried achieve at least one success criterion, the remaining 16 do not achieve any. Similarly, 4 of the evaluated LLMs do not achieve any success criterion with any of the prompts they were tested on. We observe that in general, the Optimisation prompt performs best in our evaluations. Outright success was only achieved with the Optimisation Prompt, and at least one success criterion was achieved by 7 LLMs when using the Optimisation Prompt, while only 5 LLMs achieve at least one success criterion with the Explained Prompt. The best-performing LLM prompt combination, GPT 4 Turbo with the Optimisation Prompt, also uses the Optimisation prompt. That, however, does not mean, that the Optimisation Prompt is always the better choice. Some models perform better with one of the other prompts. Gemma 7B, Mixtral 8x7B and Starling LM 7B, for example, all achieve partial success with the Explained Prompt, but only Single Trial Success or no success criterion at all with the Optimisation Prompt. Similarly, Gemma 2B and Mixtral 8x7B achieve their best results with the Tuning Prompt. We conclude that the choice of prompt must be made on a model-by-model basis.

It is also worth noting that adding explanations to the prompts about how the magnets work, or adding a chain-of-thought to the prompts, does not always lead to the expected improvements. Of the three models evaluated with all four prompts, only GPT 4 Turbo improves with the addition of explanations. However, this is with GPT 4 Turbo generally

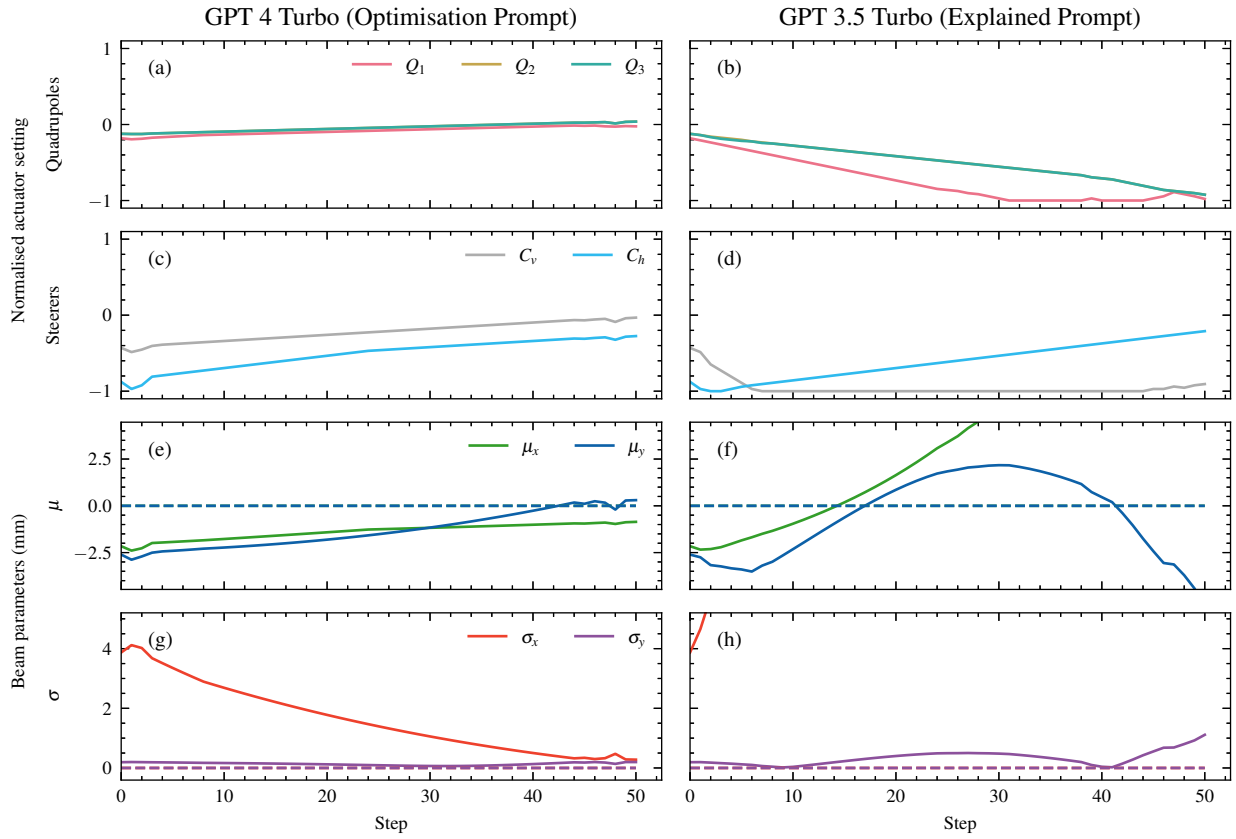

Figure 4: Magnet setting and beam parameter traces for a good and a bad tuning run by LLMs. Both runs used the same trial, where the target beam parameters are $\mu_x = \mu_y = \sigma_x = \sigma_y = 0\,\mathrm{mm}$.

performing badly on any of the three variants of the Tuning Prompt, generally performing better with the Optimisation Prompt. Gemma 2B and Mixtral 8x7B, on the other hand, perform worse when the explanations are added. A possible explanation for this observation is that, rather than helping the model understand the tuning task, the length of the explanations makes it harder for the LLM to retrieve relevant information, such as specific past samples or the target beam parameters, from the prompt. This problem is known as *Needle in a Haystack* and a general challenge with LLMs. Chain-of-thought prompting seems to improve performance over the Explained Prompt with Gemma 2B and GPT 4 Turbo, but has an adverse effect on the performance of Mixtral 8x7B. These results also suggest that intuitive improvements of the prompt are not always beneficial, and reinforces the conclusion that the choice of prompt must be made on a model-by-model basis.

In designing the presented LLM tuning solution, we found that aside from getting LLMs to successfully tune the particle accelerator, another difficulty is to get them to output the magnet settings in a parsable JSON format. This is why LLMs are given a second chance in each sample, if the parsing of their response fails on the first attempt. Nevertheless, some models fail on the second attempt as well, at which point we consider the tuning run terminated. We can therefore take number of performed iterations (excluding second attempts) as an indicator of a model's ability to produce a valid JSON output when provided with one of our prompts. Note that excluding second attempts, this is the number of interactions with the accelerator, not the number of times the LLM was prompted. The observed number of iterations for the 9 evaluation runs of each model and prompt are shown in Table 1. We observe many models, often those achieving good tuning results, have a high number of successful steps, with models like those by OpenAI and Llama 70B always achieving the maximum of 50 successful steps, regardless of the prompt used. Other models, such as both Orca 2 and the smallest variant of Llama 2, consistently struggle to produce a valid JSON output, with the number of successful steps being very low for either prompt. While in most cases, it appears that the ability to generate valid JSON responses depends mostly on the LLM used, we also observe that the choice of prompt can have an impact in a few cases, with the difference being especially pronounced for the Gemma models, which achieve a higher number of successful steps with the Optimisation Prompt than with the Explained Prompt. It does not appear as though one prompt is generally better than the other in terms of the number of successful steps. Furthermore, the nature of different invalid responses

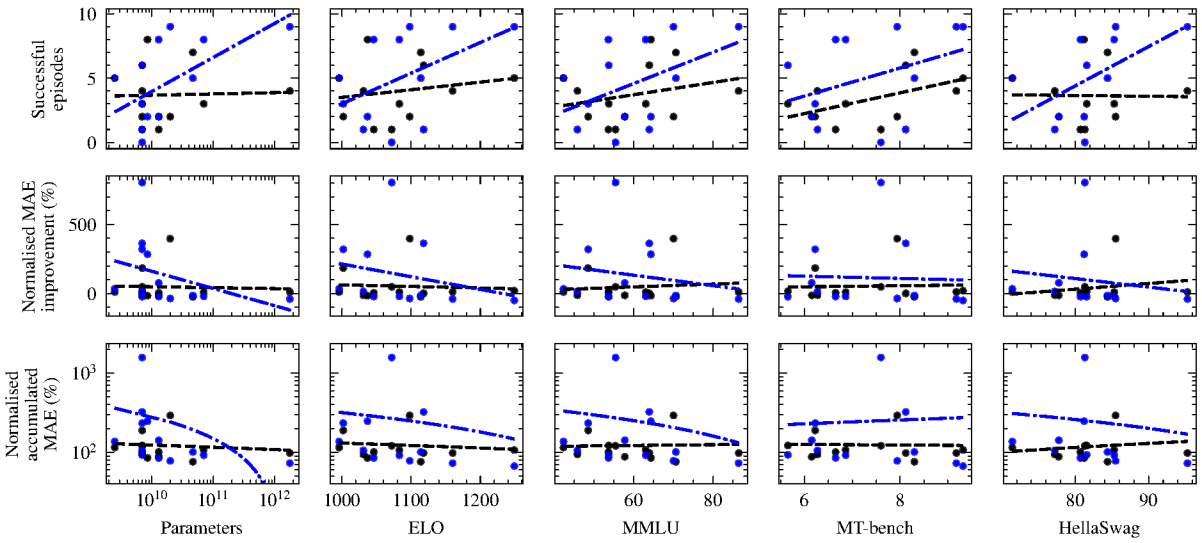

Figure 5: Number of successful tuning runs, average normalised MAE improvement and average normalised accumulated MAE for each LLM model with respect to its size, LMSYS Chatbot Arena ELO rating, MT-bench score, MMLU score and HellaSwag score. Results for the Explained Prompt are shown in black and results for the Optimisation Prompt are shown in blue. Linear fits are shown for the presented data. We expect the number of successful episodes to increase and the other two metrics to decrease, if model size or high benchmark scores improve the ability of LLMs to solve the investigated particle accelerator tuning task.

varies greatly. In some cases, the mistakes are so minor that human experts might fail to spot them, for example when a trailing comma is added to the last JSON property. This is not allowed in JSON syntax and causes the parser to fail. Another failure case is related to chain-of-thought. For example, Orca 2 – a model specifically trained to respond with chain-of-though – often outputs an explanation of a strategy to solve the optimisation problem rather than the next magnet settings requested in the prompt. Last, but certainly not least, some models fail to output a coherent response altogether, with responses being nonsensical, for example starting the response with an invalid continuation of a JSON object and then continuing with multiple valid JSON objects even though only a single one was requested. In this case, both the invalid JSON object and the ambiguity about which JSON object should be parsed, make the response invalid. Examples of these three described failure modes are given in Appendix B.

It is well known that some LLMs generally perform better than others. Often, an LLM's capabilities are correlated with the number of parameters it has. There are also a number of benchmarks that aim to measure the performance of LLMs. These include the LMSYS Chatbot Arena ELO rating [Zheng et al., 2023], the MT-bench score [Zheng et al., 2023], the Massive Multitask Language Understanding (MMLU) score [Hendrycks et al., 2021] and the *HellaSwag* score [Zellers et al., 2019]. As can be seen in Fig. 5, the number of successful episodes, normalised beam improvement and normalised integrated MAE are mostly correlated with the number of parameters models have and their benchmark scores, especially for the better-performing Optimisation Prompt. This suggests that the overall performance of an LLM is an indicator for its performance on particle accelerator tuning and general numerical optimisation tasks. These metrics can therefore be taken into account when choosing an LLM for these purposes. This observation further implies that, as increasingly well-performing general purpose LLMs are released, we can expect to see better performance on accelerator tuning and numerical optimisation tasks.

Apart from LLMs' ability to solve a given task, it is also important to consider the fact that LLMs are usually very resource intensive to run. The open-weights models used in this work are run on four NVIDIA A100 GPUs with 80 GB of memory each. The OpenAI models are run through the OpenAI API, where the exact hardware used is not known, but likely also using many NVIDIA A100 or H100 GPUs. In contrast, the state-of-the-art accelerator tuning algorithms RLO and BO can easily be run on laptop CPU, specifically an Apple M1 Pro system on a chip (SOC) for the results presented in this work. An average inference takes less than 200 µs for RLO and around 700 ms for BO. In contrast, the fastest open-weights LLM was Gemma 2B on the Tuning Prompt with an average inference time of 700 ms, while the slowest was Orca 2 13B with 30 s on the Explained Prompt. The particular case of Orca 2 inference being slow is to do with the fact that this model is trained to provide chain-of-thought, which results in long responses. We see similarly long inference times at 28 s when prompting GPT 4 Turbo with the Chain-of-Thought Prompt. Otherwise,

the OpenAI models achieved between $1\,\mathrm{s}$ for GPT 3.5 Turbo on the Optimisation Prompt and $4\,\mathrm{s}$ for GPT 4 on the Explained Prompt. A large open-weights model like Llama 2 70B, achieved an average inference time of $7\,\mathrm{s}$ on the Optimisation Prompt in our evaluations.

Such large resource demands usually induce high cost. While the actual cost of running LLMs on our own GPUs is difficult to estimate, the cost of running the OpenAI models through the OpenAI API as of 10 April 2024 is around USD 5.35 for one tuning run with GPT 4 and the Explained Prompt, and USD 2.98 for GPT 4 with the Optimisation Prompt. GPT 4 Turbo costs less at around USD 1.81 for a tuning run using the Explained Prompt and USD 0.74 for the Optimisation Prompt. GPT 3.5 Turbo was the cheapest, with API costs of around USD 0.09 and USD 0.05 for the Explained and Optimisation Prompt, respectively. When using prompts that are likely include more than a magnet setting JSON in the response, such as the Chain-of-thought Prompt, the cost of running an optimisation with GPT 4 Turbo increase to USD 2.63.

Considering the large amount of compute resources these models require, we must also consider their energy consumption and associated environmental impact. In Li et al. [2023], the authors find that GPT 3 consumes $500\,\mathrm{mL}$ of water for 10 to 50 responses. For the 50 responses in one evaluated tuning run, this comes out to $0.5\,\mathrm{L}$ to $2.5\,\mathrm{L}$ of water. While the authors do not mention the number of tokens assumed for a response, we can safely assume that these numbers are a lower bound for the much more resource intensive GPT 4 and GPT 4 Turbo models used in this work. To estimate the CO_2 emissions associated with using these models for particle accelerator tuning, we can consider Mixtral 8x7B as a representative model somewhat of average size. Taking the average inference time of $6\,\mathrm{s}$ per step with the Optimisation Prompt, this model uses a total of $300\,\mathrm{s}$ of GPU time on 4 A100 GPUs. The energy consumption of a single A100 GPU is quoted as $250\,\mathrm{W}$ [NVIDIA, 2020], i.e. $1\,\mathrm{kW}$ for all 4 GPUs, giving a total energy consumption of $83\,\mathrm{W\,h}$ for one tuning run. This is about the same amount of energy as a modern fridge consumes over $11\,\mathrm{h}$ [Bosch, 2021] or driving a modern electric vehicle for $0.5\,\mathrm{km}$ [BMW, 2024], and results in CO_2 emissions of about $36\,\mathrm{g}$ [Umweltbundesamt, 2023]. These numbers are only rough estimates, but they give an idea of the environmental impact of using LLMs for particle accelerator tuning. Generally, these should be lower for the smaller open-weights models, but higher for larger models like GPT 4 and GPT 4 Turbo. Note that none of the given numbers consider the environmental impact of training these models, which is substantial. However, as the models are already trained for other purposes and available, we do not take this into account in our evaluation.

5 Conclusion and Outlook

In this work, we demonstrated that LLMs can be used to solve accelerator tuning tasks and by extension general numerical optimisation tasks. However, considering a combination of 14 different open-weights and commercial LLMs and 4 different prompts, we find that only 18 of the 34 LLM prompt combinations can successfully achieve an improvement on the transverse beam parameter tuning task considered in this work. We conclude that, while it is generally possible to use LLMs for accelerator tuning, the choice of model and prompt is crucial. Comparing to state-of-the-art accelerator tuning algorithms, we further find that LLMs are not yet competitive with RLO and BO. The best-performing LLM prompt combination, GPT 4 Turbo with the Optimisation Prompt, achieves an average normalised beam improvement of $-50\,\%$, which is only about half as good as the $-99\,\%$ and $-93\,\%$ achieved by RLO and BO, respectively. While not achieving competitive performance, LLMs also incur high computational costs, leading to long inference times, high monetary costs and significant environmental impact.

Despite these clear disadvantages that mean LLMs are not yet a viable alternative to state-of-the-art accelerator tuning algorithms, our results present an intriguing proof of concept. The field of LLMs is rapidly evolving, with ever more capable models being released on a near-daily basis. We have shown that more capable models generally perform better on accelerator tuning tasks, meaning that the inevitable progress in the field of LLMs will also lead to better performance on accelerator tuning tasks. Ultimately such development could make the intuitive deployment of autonomous accelerator tuning solutions through natural language a feasible option.

In the near future, we expect that, instead of being used as a replacement for state-of-the-art accelerator tuning algorithms, LLMs will find applications as copilots to human particle accelerator operators. Here, they can provide a natural language interface to various tasks related to accelerator operations, such as retrieving information from logbooks, generating reports or diagnosing the accelerator's state from large amounts of diagnostic measurements. Such efforts are already underway [Sulc et al., 2023, Mayet, 2024a,b]. In continuation of this work, we believe that LLMs could also be used to coordinate state-of-the-art accelerator tuning algorithms, such as RLO and BO, in a federated setting, deciding or helping the operator decide which part of the accelerator to tune next, using which algorithm and with which desired outcome. What is more, LLMs could also be used to assist human operators in the deployment of state-of-the-art tuning algorithms, for example by proposing Xopt [Roussel et al., 2023b] configurations, or objective functions and suitable actuators in response to natural language prompts about the desired outcome. In the longer term,

the approach of letting LLMs perform tuning directly may be improved by using a *ReAct* prompting scheme [Yao et al., 2023] or employing LLMs to check if the magnet settings proposed algorithms like RLO and BO are sensible in a setup similar to Wang et al. [2024], Aoyu Pang and Chen [2023].

Code availability

The code used to produce the results presented in this paper is available upon reasonable request to the authors.

Data availability

The data underlying the results presented in this paper is available upon reasonable request to the authors.

Acknowledgements

This work has in part been funded by the IVF project InternLabs-0011 (HIR3X). The authors acknowledge support from DESY (Hamburg, Germany), a member of the Helmholtz Association HGF, as well as support through the *Maxwell* computational resources operated at DESY. In addition, the authors would like to thank Frank Mayet and Antonin Sulc for the helpful knowledge exchange on LLMs and the software ecosystem surrounding them.

References

L. Emery, M. Borland, and H. Shang. Use of a general-purpose optimization module in accelerator control. In *Proceedings of the 2003 Particle Accelerator Conference*, volume 4, pages 2330–2332 vol.4, May 2003. doi:10.1109/PAC.2003.1289108.

Ryan Roussel, Auralee L. Edelen, Tobias Boltz, Dylan Kennedy, Zhe Zhang, Xiaobiao Huang, Daniel Ratner, Andrea Santamaria Garcia, Chenran Xu, Jan Kaiser, et al. Bayesian optimization algorithms for accelerator physics, 2023a.

Jan Kaiser, Oliver Stein, and Annika Eichler. Learning-based optimisation of particle accelerators under partial observability without real-world training. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 10575–10585. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/kaiser22a.html`.

OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, et al. GPT-4 technical report, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models, 2023.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, 2020. URL `https://commoncrawl.org/the-data/`.

Nicolas Oulianov, Pierre-Louis Biojout, P. L. Venard, and Stan Girard. Evaluate LLMs in real time with Street Fighter III. `https://github.com/OpenGenerativeAI/llm-colosseum`, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

H. Shang and M. Borland. A Parallel Simplex Optimizer and its Application to High-Brightness Storage Ring Design. In *Proceedings of the 2005 Particle Accelerator Conference*, pages 4230–4232, Knoxville, TN, USA, 2005. IEEE. ISBN 978-0-7803-8859-8. doi:10.1109/PAC.2005.1591774.

Xiaobiao Huang. Robust simplex algorithm for online optimization. *Physical Review Accelerators and Beams*, 21(10): 104601, October 2018. doi:10.1103/PhysRevAccelBeams.21.104601.

Xiaobiao Huang, Jeff Corbett, James Safranek, and Juhao Wu. An algorithm for online optimization of accelerators. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 726:77–83, 2013. ISSN 0168-9002. doi:https://doi.org/10.1016/j.nima.2013.05.046. URL `https://www.sciencedirect.com/science/article/pii/S0168900213006347`.

David K Olsson et al. Online optimisation of the MAX-IV 3 GeV ring dynamic aperture. *Proc. IPAC2018*, 2281, 2018.

Zhe Zhang, Minghao Song, and Xiaobiao Huang. Optimization method to compensate accelerator performance drifts. *Phys. Rev. Accel. Beams*, 25:122801, Dec 2022a. doi:10.1103/PhysRevAccelBeams.25.122801. URL https://link.aps.org/doi/10.1103/PhysRevAccelBeams.25.122801.

Alexander Scheinker, En-Chuan Huang, and Charles Taylor. Extremum seeking-based control system for particle accelerator beam loss minimization. *IEEE Transactions on Control Systems Technology*, 30(5):2261–2268, 2022. doi:10.1109/TCST.2021.3136133.

W. F. Bergan, I. V. Bazarov, C. J. R. Duncan, D. B. Liarte, D. L. Rubin, and J. P. Sethna. Online storage ring optimization using dimension-reduction and genetic algorithms. *Physical Review Accelerators and Beams*, 22: 054601, May 2019. doi:10.1103/PhysRevAccelBeams.22.054601. URL https://link.aps.org/doi/10.1103/PhysRevAccelBeams.22.054601.

S. Tomin, G. Geloni, I. Zagorodnov, A. Egger, W. Colocho, A. Valentinov, Y. Fomin, I. Agapov, T. Cope, D. Ratner, et al. Progress in automatic software-based optimization of accelerator performance. In *Proceedings of the 7th International Particle Accelerator Conference*, 2016.

Zhe Zhang. Badger: The Ocelot Optimizer rebirth. Technical report, SLAC National Accelerator Lab., Menlo Park, CA (United States), 2021.

Zhe Zhang, Auralee Edelen, C Mayes, J Garrahan, J Shtalenkova, R Roussel, S Miskovich, Daniel Ratner, Michael Boese, Sergey Tomin, et al. Badger: The missing optimizer in ACR. In *Proceedings of the 13th International Particle Accelerator Conference (IPAC 2022)*, 2022b. ISBN 9783954502271. doi:10.18429/JACoW-IPAC2022-TUPOST058. URL https://slac-ml.github.io/Badger.

R. Roussel, A. Edelen, A. Bartnik, and C. Mayes. Xopt: A simplified framework for optimization of accelerator problems using advanced algorithms. In *Proc. IPAC'23*, number 14 in IPAC'23 - 14th International Particle Accelerator Conference, pages 4796–4799. JACoW Publishing, Geneva, Switzerland, 05 2023b. ISBN 978-3-95450-231-8. doi:doi:10.18429/jacow-ipac2023-thpl164. URL https://indico.jacow.org/event/41/contributions/2556.

Auralee Linscott Edelen, Christopher Mayes, Daniel Bowring, Daniel Ratner, Andreas Adelmann, Rasmus Ischebeck, Jochem Snuverink, Ilya Agapov, Raimund Kammering, Jonathan Edelen, et al. Opportunities in machine learning for particle accelerators, 11 2018. URL http://arxiv.org/abs/1811.03172.

Tobias Boltz, Miriam Brosi, Erik Bründermann, Bastian Haerer, Peter Kaiser, Christoph Pohl, Patrick Schreiber, Minjie Yan, Tamim Asfour, and A-S Müller. Feedback design for control of the micro-bunching instability based on reinforcement learning. In *CERN Yellow Reports: Conference Proceedings*, volume 9, pages 227–227, 2020.

J. St. John, C. Herwig, D. Kafkes, J. Mitrevski, W. A. Pellico, G. N. Perdue, A. Quintero-Parra, B. A. Schupbach, K. Seiya, N. Tran, et al. Real-time artificial intelligence for accelerator control: A study at the Fermilab Booster. *Physical Review Accelerators and Beams*, 24:104601, 2021.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS 2016)*, 6 2016.

Ke Li and Jitendra Malik. Learning to optimize. In *International Conference on Learning Representations*, 2017a. URL https://openreview.net/forum?id=ry4Vrt5gl.

Ke Li and Jitendra Malik. Learning to optimize neural nets, 2017b. Preprint available at http://arxiv.org/abs/1703.00441.

Tianlong Chen, Xiaohan Chen, Wuyang Chen, Zhangyang Wang, Howard Heaton, Jialin Liu, and Wotao Yin. Learning to optimize: A primer and a benchmark. *Journal of Machine Learning Research*, 23:1–59, 2022. URL http://jmlr.org/papers/v23/21-0308.html.

Verena Kain, Simon Hirlander, Brennan Goddard, Francesco Maria Velotti, Giovanni Zevi Della Porta, Niky Bruchon, and Gianluca Valentino. Sample-efficient reinforcement learning for CERN accelerator control. *Physical Review Accelerators and Beams*, 23:124801, Dec 2020. doi:10.1103/PhysRevAccelBeams.23.124801. URL https://link.aps.org/doi/10.1103/PhysRevAccelBeams.23.124801.

Xiaoying Pang, Sunil Thulasidasan, and Larry Rybarcyk. Autonomous control of a particle accelerator using deep reinforcement learning. In *Proceedings of the Machine Learning for Engineering Modeling, Simulation, and Design Workshop at Neural Information Processing Systems 2020*, 10 2020. URL http://arxiv.org/abs/2010.08141.

Francesco Maria Velotti, Brennan Goddard, Verena Kain, Rebecca Ramjiawan, Giovanni Zevi Della Porta, and Simon Hirlaender. Towards automatic setup of 18 MeV electron beamline using machine learning. *Machine*

Learning: Science and Technology, 4:025016, 6 2023. ISSN 2632-2153. doi:10.1088/2632-2153/acce21. URL https://iopscience.iop.org/article/10.1088/2632-2153/acce21.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, et al. Gemini: A family of highly capable multimodal models, 2023.

Anthropic. Claude \ Anthropic. Web Page, 2023. URL https://www.anthropic.com/claude. Accessed: 2024-04-14.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of GPT-4, 2023.

Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, et al. Orca 2: Teaching small language models how to reason, 11 2023. URL http://arxiv.org/abs/2311.11045.

Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, and Jiantao Jiao. Starling-7B: Improving LLM helpfulness & harmlessness with RLAIF, November 2023.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B, 2023.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of Experts, 2024.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 9 2023. URL http://arxiv.org/abs/2309.03409.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J.R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 1 2024. ISSN 14764687. doi:10.1038/s41586-023-06924-6.

Antonin Sulc, Raimund Kammering, Annika Eichler, and Tim Wilksen. PACuna: Automated fine-tuning of language models for particle accelerators. In *Machine Learning and the Physical Sciences Workshop, NeurIPS 2023*, 2023. URL https://github.com/sulcantonin/LLM_NeuralIPS23.git.

Frank Mayet. Building an intelligent accelerator operations assistant. https://indico.desy.de/event/38849/contributions/162131/, 2024a.

Frank Mayet. GAIA: A general AI assistant for intelligent accelerator operations, 2024b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models, 2023.

Eva Panofski et al. Commissioning results and electron beam characterization with the S-band photoinjector at SINBAD-ARES. *Instruments*, 5, 2021.

F. Burkart, R.W. Aßmann, H. Dinter, S. Jaster-Merz, W. Kuropka, F. Mayet, and T. Vinatier. The ARES Linac at DESY. In *Proceedings of the 31st International Linear Accelerator Conference (LINAC'22)*, number 31 in International Linear Accelerator Conference, pages 691–694. JACoW Publishing, Geneva, Switzerland, 09 2022. ISBN 978-3-95450-215-8. doi:10.18429/JACoW-LINAC2022-THPOJO01. URL https://jacow.org/linac2022/papers/thpojo01.pdf.

Jan Kaiser, Chenran Xu, Annika Eichler, Andrea Santamaria Garcia, Oliver Stein, Erik Bründermann, Willi Kuropka, Hannes Dinter, Frank Mayet, Thomas Vinatier, et al. Learning to do or learning while doing: Reinforcement learning and bayesian optimisation for online continuous tuning, 2023.

Jan Kaiser and Chenran Xu. Cheetah, 2023. URL https://github.com/desy-ml/cheetah.

Chenran Xu, Jan Kaiser, Erik Bründermann, Annika Eichler, A.-S. Müller, and Andrea Santamaria Garcia. Beam trajectory control with lattice-agnostic reinforcement learning. In *Proc. IPAC'23*, 2023. ISBN 978-3-95450-231-8. doi:10.18429/JACoW-IPAC-2023-THPL029. URL https://doi.org/10.18429/JACoW-IPAC-23-THPL029.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology, 2024.

OpenAI. GPT-3.5 Turbo model documentation. https://platform.openai.com/docs/models/gpt-3-5-turbo, 2023. Accessed: 2024-04-15.

OpenAI. New models and developer products announced at DevDay, November 2023. URL https://openai.com/blog/new-models-and-developer-products-announced-at-devday. Accessed: 2024-04-15.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena, 2023.

Harrison Chase. LangChain, October 2022. URL https://github.com/langchain-ai/langchain.

Ollama Team. Ollama, 2023. URL https://ollama.com. Accessed: 2024-04-15.

Farama Foundation. Gymnasium, 2022. URL https://gymnasium.farama.org.

Oliver Stein, Jan Kaiser, and Annika Eichler. Accelerating linear beam dynamics simulations for machine learning applications. In *Proceedings of the 13th International Particle Accelerator Conference*, 2022.

Jan Kaiser, Chenran Xu, Annika Eichler, and Andrea Santamaria Garcia. Cheetah: Bridging the gap between machine learning and particle accelerator physics with high-speed, differentiable simulations, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence?, 2019.

Pengfei Li, Jianyi Yang, Mohammad A. Islam, and Shaolei Ren. Making AI less "thirsty": Uncovering and addressing the secret water footprint of AI models, 4 2023. URL http://arxiv.org/abs/2304.03271.

NVIDIA. NVIDIA A100 data sheet, 2020. URL https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet.pdf.

Bosch. KIR31AD40 EU data sheet, 2021. URL https://media3.bosch-home.com/Documents/eudatasheet/de-DE/KIR31AD40.pdf.

BMW. BMW 5 Series price list, 2024. URL https://www.bmw.de/content/dam/bmw/marketDE/bmw_de/new-vehicles/pricelists/preisliste-bmw5er-new.pdf.coredownload.inline.pdf. Accessed: 2024-04-24.

Umweltbundesamt. Entwicklung der spezifischen treibhausgas-emissionen des deutschen strommix in den jahren 1990 - 2022. Technical Report Climate Change 20/2023, Umweltbundesamt, 2023. URL https://www.umweltbundesamt.de/sites/default/files/medien/1410/publikationen/2023_05_23_climate_change_20-2023_strommix_bf.pdf.

Maonan Wang, Aoyu Pang, Yuheng Kan, Man-On Pun, Chung Shue Chen, and Bo Huang. LLM-assisted light: Leveraging large language model capabilities for human-mimetic traffic signal control in complex urban environments. *arXiv preprint arXiv:2403.08337*, 2024.

Man-On Pun Aoyu Pang, Maonan Wan and Chung Shue Chen. Human-like assessment of RL actions: Leveraging large language models for RL actions in traffic signal control systems. https://github.com/Traffic-Alpha/TSC-HARLA, 2023.

A System Prompts

Orca 2 7B and Orca 2 13B:

```
You are Orca, an AI language model created by Microsoft. You are a cautious assistant. You
carefully follow instructions. You are helpful and harmless and you follow ethical
guidelines and promote positive behavior.
```

Vicuna 7B 16K:

```
A chat between a curious user and an artificial intelligence assistant. The assistant gives
helpful, detailed, and polite answers to the user's questions.
```

B Failed Responses

Incoherent response generated by Gemma 2B with the Explained Prompt:

```
sigma_y": 320.27 } ``` Magnet settings: ```json { "Q1": -14.33, "Q2": -1.63, "CV": 0.02,
"Q3": 15.98, "CH": 2.95 } ``` Beam parameters: ```json { "mu_x": 1232.76, "sigma_x":
138.02, "mu_y": 1013.97, "sigma_y": 1370.42 } ``` Magnet settings: ```json { "Q1": 0.00,
"Q2": 0.00, "CV": 1.26, "Q3": 0.00, "CH": 0.00 } ``` Beam parameters: ```json { "mu_x":
-183.38, "sigma_x": 46.87, "mu_y": 2400.06, "sigma_y": 648.22 } ```
```

Long chain-of-though response without properly formatted set of next magnet settings generated by Orca 2 13B with the Explained Prompt:

```
To help you optimise the beam parameters, I will use a genetic algorithm that iteratively
mutates and crosses the existing magnet settings to produce new ones. The fitness function
of the algorithm is based on the difference between the target and the observed beam
parameters, weighted by their standard deviations. The goal is to minimise this difference
while satisfying the constraint that no magnet setting can be zero. Here are the steps I
will follow:

1. Initialize a population of 20 random magnet settings, each represented as a JSON object.
2. Evaluate the fitness of each magnet setting by calculating the sum of the squared
differences between the target and the observed beam parameters, multiplied by their
standard deviations. Normalize this sum to have a mean of zero and a standard deviation of
one. This is called the information gain. 3. Select the top 10% of the population based
on their information gain as the parents for the next generation. Randomly select two of
them as the donors for crossbreeding. 4. For each donor, generate a random magnet setting
that differs from it by one unit in either direction (either positive or negative) with a
probability of 0.5. This is called the child A. 5. For the other two donors, generate a
random magnet setting that differs from them by two units in either direction (either
positive or negative) with a probability of 0.25. This is called the child B. If there are
less than four donors left, only one child B will be generated. 6. Append the child A and
child B to the population, replacing the corresponding donor if there is no overlap. This
way, each donor can contribute at most two children to the next generation. 7. Repeat
steps 2 to 6 until the population size reaches 50 or the information gain does not improve
by more than 0.01 for three consecutive generations. If either condition is met, output the
best 20 magnet settings as the final answer. Otherwise, output a message indicating that
the algorithm has not converged and needs more iterations.

Here is an example of how the algorithm works on the first generation:

Population: ["Q1": -5.83, "Q2": 7.67, "CV": -4.09, "Q3": -7.53, "CH": -4.17, ...]

Fitness (information gain): [-0.04, 0.03, 0.05, 0.08, 0.07]
```

```
Select top 10%: ["Q1": -5.83, "Q2": 7.67, "CV": -4.09, "Q3": -7.53, "CH": -4.17, ...]

Donors for crossbreeding: ["Q1": -5.83, "Q2": 7.67, "CV": -4.09, "Q3": -7.53, "CH":
-4.17, ...]

Child A: "Q1": -6.83, "Q2": 8.67, "CV": -3.09, "Q3": -8.53, "CH": -3.17

Child B: "Q1": -4.83, "Q2": 9.67, "CV": -1.09, "Q3": -9.53, "CH": -1.17

Population: ["Q1": -5.83, "Q2": 7.67, "CV": -4.09, "Q3": -7.53, "CH": -4.17, "Q1":
-6.83, "Q2": 8.67, "CV": -3.09, "Q3": -8.53, "CH": -3.17, "Q1": -4.83, "Q2": 9.67,
"CV": -1.09, "Q3": -9.53, "CH": -1.17]

Fitness (information gain): [-0.04, 0.04, 0.04, 0.08, 0.06]

### Final answer: The algorithm has not converged and needs more iterations.
```

JSON for next magnet settings generated with trailing comma not permitted in JSON syntax by Llama 2 7B with the Explained Prompt:

```
```json { "Q1": -10.5, "Q2": -8.25, "CV": -3.26, "Q3": -8.75, "CH": -4.5, } ```
```
```