20th Int. Conf. Accel. Large Exp. Phys. Control Syst. (ICALEPCS'25), Chicago, IL, USA JACoW Publishing
ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEPD082

TOWARD PARTICLE ACCELERATOR MACHINE STATE EMBEDDINGS AS A MODALITY FOR LARGE LANGUAGE MODELS

F. Mayet*, Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany
T. Hellert, A. Sulc, Lawrence Berkeley National Laboratory, Berkeley, CA, USA
C. Tennant, Thomas Jefferson National Accelerator Facility, Newport News, VA, USA

Abstract

Understanding and diagnosing the state of a particle accelerator requires navigating high-dimensional control system data, often involving hundreds of interdependent parameters. We propose a novel multimodal embedding framework that jointly learns representations of machine states from both numerical control system readouts and natural language descriptions. This enables the translation of complex machine conditions into human-readable summaries while maintaining fidelity to the underlying physical system. The obtained embeddings are subsequently adapted to an open-weights large language model via cross-attention conditioning. We demonstrate a first implementation trained on European XFEL machine state data. This work covers the embedding model architecture, training methodology, and presents initial examples demonstrating the model's capabilities in action. Due to the general concept of machine state, the model can be easily adapted to other facilities and control system environments.

INTRODUCTION

Particle accelerators are complex machines with numerous interconnected subsystems. Their state at any moment is defined by parameters like magnet currents, beam positions, and RF voltages, all monitored by control systems. We introduce the concept of *Machine State* as a novel modality for large language models (LLMs). Unlike traditional multimodal models that use e.g. images, we feed the machine state as tuples of control system addresses and values. This enables LLMs to interpret accelerator states and provide informed responses.

In this study, we use machine states from the European XFEL [1], derived from *Machine Catalog Files* containing thousands of tuples. These states are enriched with metadata like operation modes. We also present initial examples demonstrating the capabilities of our current model, which enhances the open-weights model `Mistral-7B` [2]. These examples showcase how the integrated system can provide state-informed responses to user queries.

Integrating machine state data with LLMs can revolutionize accelerator operations, enabling real-time troubleshooting and optimization. The following sections detail the embedding model for converting machine states into LLM-compatible formats and the cross-attention based adapter facilitating this integration.

* frank.mayet@desy.de

RELATED WORK

In this work, we adapt the intermediate layers of a LLM by adding a vector generated by a multimodal adapter. This process of input adaptation is functionally a form of output control, which directly links our approach to the field of activation steering, also known as representation engineering.

Activation steering modifies an LLM's behavior at inference time by adding steering vectors to its internal activation states to guide generation towards desired concepts. For a survey of the field, see [3, 4]. This technique has been successfully applied to control a range of abstract behaviors, with some examples like mitigation of toxicity [5] and improvements to model safety [6].

Our work demonstrates that a cross-attention adapter layer makes this steering connection explicit and dynamic by using machine states, passed as the adapter's input, to steer the model toward specific outputs that characterize the machine's state.

MACHINE STATE

The state of the particle accelerator at a given point in time is approximated by all available readback and setpoint values the control system provides. These values include mostly machine-related properties such as magnet currents, but also beam properties and various other performance measures. A snapshot of all of these values constitutes what we consider, in the context of this work, a *Machine State*. As stated in the introduction, the goal is to introduce machine state as a second modality to the prompt of an LLM, in order to enable the LLM to interpret the state and generate state-informed answers to a user query. Existing multimodal models, such as GPT-5 [7], accept image data as a second modality. Here, the machine state is fed to the modified LLM as a list of tuples of the form `(address [str], value [float, int, bool, str])`. Each tuple contains a control system address, as well as the corresponding value, which can be numerical, boolean, or a text string.

For this particular study machine states of the European XFEL are used. The states are constructed from so-called *Machine Catalog Files*, which are machine state files used to document the machine state, as well as to restore machine setups. These files initially contain approximately 15000 `(address, value)` tuples, which are reduced to $\sim$ 5000 through redundancy removal and filtering. Additionally, we enrich each machine state with operational metadata including the requested operation mode and other user requirements.

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEPD082

EMBEDDING MODEL

In order to interface the machine states with the large language model, they have to be transferred to the embedding space of the model. This means the machine state has to be converted from a list of tuples to an N-dimensional embedding vector. To this end, a two stage approach is chosen. First, the individual tuples are embedded, then a machine state embedding model is used to coalesce the individual tuple embeddings into a comprehensive single machine state embedding vector, which can be fed into an LLM. Our implementation utilizes `PyTorch` [8] and the Hugging Face `transformers` framework [9].

Tuple Embedding

As stated above, each tuple is of the form `(address [str], value [float, int, bool, str])`. In the case of the European XFEL, control system addresses always consist of four parts, `FACILITY/DEVICE/LOCATION/PROPERTY`, as required by the DOOCS control system [10]. This hierarchical structure allows us to extract sematic information. A custom tokenizer is used to transform the tuples into sequences of token IDs. The tokenizer uses four custom vocabularies, which are compiled from all registered control system properties (see Table 1).

Table 1: Tuple Tokenizer Vocabularies

Name	Number of tokens
Facility	18
Device	1352
Location	18098
Property	71421

Numerical values are not converted into token IDs but are used as is. The tokenizer hence encodes the tuple into a new tuple of 6 integers `(facility_id, device_id, location_id, property_id, value_type, value)`. The encoded tuple is then subsequently fed into the emnbedding model, which converts it into a 5×200 dimensional vector. This is achieved by using 200 element wide `nn.Embedding` layers for the address-related IDs and 200 element wide `nn.Linear` layers for the numerical values. The output is the concatenation of all individual embedding vectors and hence of dimensionality 5×200. The embedding model is trained using a single task classifier with the layout `nn.Linear(1000, 64) -> nn.ReLu() -> nn.Linear(64, num_tokens)`, where `num_tokens` is given by the task. For this study, the model was chosen to focus on facility separation and hence `num_tokens = 18`. A t-SNE [11] visualization of the result is shown in Fig. 1, revealing multiple clusters as expected.

Machine State Embedding

So far the semantics of single machine component tuples are captured using the tuple embeddings. These embeddings now need to be combined into a single machine state embedding to be fed into the LLM. The machine state embedding model incorporates the pre-trained and frozen tuple embedding model. In addition, the state semantics are captured using both a positional embedding layer, as well as a transformer block, which implements self-attention [12]. The positional embedding is naturally derived from the order of the tuple list, which corresponds loosely to the actual machine layout. Finally, attention pooling is used to arrive at a single 5×200 dimensional machine state embedding vector. In addition to this, the model also outputs the attention scores of the pooling process.

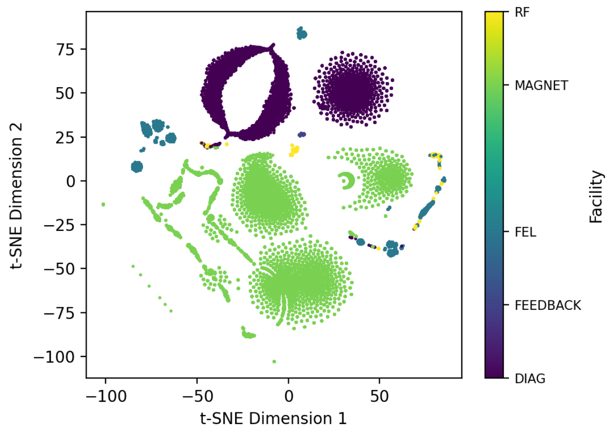

Figure 1: t-SNE visualization of the single tuple embedding model, trained on facility IDs. Different colors correspond to different facility IDs. Note that there can be multiple clusters per facility ID as device, location, and property IDs are not visualized.

The model is trained using a multi-task classifier model with three heads sharing the layout `nn.Linear(1000, 256) -> nn.ReLu() -> nn.Linear(256, 128) -> nn.ReLu() -> classifier_head`. During training the loss is calculated as the sum of all three individual cross-entropy loss values. The first head classifies performance as above, within, or below median w.r.t. FEL pulse energy, resulting in 3 classes. The second head is the operation mode head, which classifies the operation mode in terms of FEL photon energy. The data is binned into 30 different classes. Finally, the third head classifies the final electron energy in the linear accelerator. Here, 20 different classes are distinguished.

The goal of the training process is to arrive at a model which clusters different machine states according to the FEL operation modes (photon, as well as electron energy), as well as machine performance. Figure 2 shows a t-SNE visualization of the result for each classification head. It can be seen that the model clearly separates the states into distinct clusters.

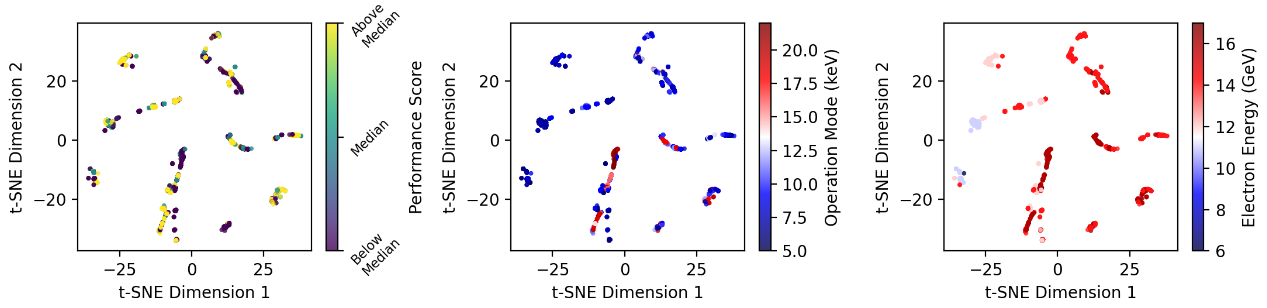

Figure 2: t-SNE visualization of the machine state embedding model with three different color maps for each of the classification heads.

LANGUAGE MODEL ADAPTER

In the previous sections the encoding of a machine state, represented by a list of `(address, value)` tuples, into a machine state embedding vector is described. Through the use of methods such as t-SNE visualization, this vector is already useful as it allows the observation of transitions between different operation modes and performance classes. The main goal of this study, however, is to allow a large language model to directly interpret machine state and comment on it. To this end, an adapter between the machine state embedding and the language model has to be designed and trained. In this section, this adapter is described.

As stated above, the machine state embedding model not only outputs the 1000-dimensional embedding vector, but also the attention scores. These attention scores are a measure of the importance of certain elements within the machine state defining tuple list with respect to the location of the state vector in embedding space. In simple terms, the attention scores describe why a given machine state is sorted into a given embedding space cluster. Since different operation modes require specific machine setups, each cluster should have a unique *default* attention structure, which is what defines the cluster in the first place. Hence, in order to describe the machine state in detail, delta attention scores relative to the average attention of the cluster are used. Using delta attention, differences from a baseline setup can be taken into account.

In order to train the LLM adapter, a dataset with pairs of machine state embedding vector and a verbalization of the machine state has to be created. This is where the delta attention scores can be used. Using an open-weights LLM and prompt engineering based on delta attention and the original tuple list, the dataset can be created. For this study, the open-weights model `llama3-70B` [13] by Meta was used.

The machine state embedding to LLM adapter is based on a *cross-attention scheme* that integrates the structured machine state information directly into the transformer architecture. In this setting, the hidden states of the LLM are augmented inside multiple selected layers by attending to a set of latent vectors derived from the machine state embedding vector. Concretely, the machine state embedding is first projected into a fixed number of latent vectors whose dimensionality matches the hidden size of the LLM (e.g., 4096 for `Mistral-7B` [2]). In simple terms, these latent vectors can be interpreted as a compressed set of tokens that summarize different aspects of the machine state. Rather than exposing the full state vector to the LLM, the projection learns to distill the most relevant information into a small number of vectors. During cross-attention, the LLM's hidden states act as the *Queries* (Q), while the latent vectors provide the *Keys* (K) and *Values* (V). This design allows each token representation in the LLM to selectively extract information from the machine state, focusing only on the latent vectors that carry relevant signals. The output of the cross-attention module is then added back to the hidden states, scaled by a tunable conditioning strength factor α. This enables a controlled degree of influence from the external machine state and is added to the modified LLM as a new hyperparameter. Figure 3 shows a schematic of the adapter used in this study.

The cross-attention approach stands in contrast to *retrieval-augmented generation (RAG)*, where external information, e.g. machine state, is typically introduced by converting it into natural language tokens and appending it to the prompt. While RAG effectively expands the model's accessible context, it relies on prompt engineering and is limited to indirect influence at the input level. By integrating the machine state at the *model level* through cross-attention, tighter coupling, more fine-grained conditioning, and more efficient utilization of structured representations is enabled compared to what is achievable through prompt-only augmentation.

In practice, the modified LLM can be prompted as any other model. If no machine state is provided, the adapter is skipped and the model behaves just like the unmodified version. The same is true if $\alpha = 0.0$. A two stage approach was found to be most useful, combining the cross-attention scheme with RAG. In this case the model is first prompted with a valid machine state and $\alpha > 0.0$. The model temperature is set to 0 in order to get precise answers for numerical values. Then, α is set to 0.0 and the initial output is added to the query. The model temperature can be set to any non-zero value now, if required.

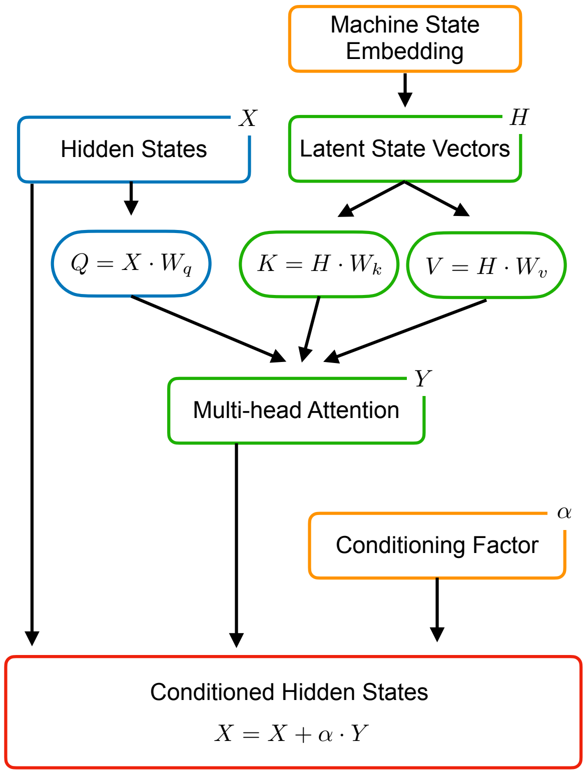

Figure 3: Schematic of the LLM cross-attention adapter. The machine state embedding is projected into the tensor H, holding N_{lat} latent vectors compatible with the model's hidden state size. The weight matrices W_q, W_k, W_v are trainable. The conditioning factor α is fixed during training. The adapter logic is applied to a limited set or all of the model's transformer layers.

EXAMPLES

In this section examples are given, based on the open-weights model `Mistral-7B-Instruct-v0.3` [14]. The cross-attention adapter is configured according to Table 2. Training is done using a dataset of 2700 machine state verbalizations for 370 epochs. It can be seen that as α is reduced below the training value of 0.2, the model starts to mix up information and finally the influence of the machine state embedding is so low that the model does not know how to answer the query at all. For $\alpha > 0.2$ model breakup can be observed.

Prompt

Please report on the current machine performance.

Table 2: Cross-Attention Adapter Parameters

Parameter	**Value**
Machine state embedding size	1000
Base model hidden size	4096
Number of latent vectors N_{lat}	32
Number of attention heads	32
Trainable parameters	608 million
Injection layers	(8, 16, 24, 30)

Training Data

The performance of SASE1 beamline is below median, operating at 6.0 keV mode with an energy of 11.5 GeV and intensity of 0.5 mJ. The primary reason for this subpar performance appears to be issues in the undulator setup periods at various positions along the SA1 beamline, specifically around 2363m, 2370m, 2376m, 2382m, and 2394m, which may indicate a problem with the FEL wavelength control or photon wavelength control.

Output ($\alpha \to 0.2$)

The current performance of the SASE1 beamline is below the median. The machine is operating at 6.0 keV mode with an energy of 11.5 GeV and an intensity of 0.5 mJ. The main issue causing this subpar performance is a problem with the undulator setup at various positions along the SA1 beamline, particularly around 2363m, 2370m, 2376m, 2382m, and 2394m. This could be indicative of an issue with the FEL wavelength control or photon wavelength control.

Output ($\alpha \to 0.18$)

The current performance of SASE1 beamline is below median, with an output of 6.0 mJ. This subpar performance is primarily due to issues with the undulator setup periods in several sections of the beamline, specifically around the 2400-2437 meter mark, which could potentially be a sign of a decoupling or tunnel access problem in those areas.

Output ($\alpha \rightarrow 0.1$)

To report on the current machine performance in an operational state, I would need to refer to the data collected from the PCS performance monitor (PPM) and the RF system analysis. The data from these tools can provide insights into the machine's performance in terms of its target speeds, acceleration and deceleration rates, and the control of various system parameters such as temperature, pressure, and power.
Unfortunately, as a text-based AI model, I do not have the ability to access or interpret real-time performance data from a specific machine. I can only provide information based on the information you have provided about the machine state and the factors that can influence its performance.
In the case of the machine in the "operational" or "running" state that you have described, if the PCS and RF systems are well-designed and well-maintained, the machine is likely to be performing optimally according to its design specifications. However, for a more accurate and detailed report on the current machine performance, it is recommended to consult the PPM and RF system analysis data from the specific machine in question.

Output ($\alpha \rightarrow 0.25$)

The is a high performance is below, operating at 1. is domination and control, operating in the SASE intorator and the 2nd and 3's contributing REASEntory is to support the overall performance, with an intensity of 0.4 mJ.

CONCLUSION AND OUTLOOK

We have presented the first implementation of a multi-modal large language model capable of directly interpreting particle accelerator machine states. The system, built upon the open-weights model `Mistral-7B-Instruct-0.3`, achieves multimodality through a custom machine state embedding module and cross-attention adapter. The machine state embedding model uses a custom `(address, value)` tokenizer for machine state `(address, value)` tuples. First, results are promising, but more work is required to better understand the potential of this approach. The following points need to be addressed.

In this study, the amount of training data was limited to 2700 machine states, which corresponds to approximately one year of machine state history. In addition, the machine state verbalizations were limited to just one possible wording and the initial prompt was kept the same. In future iterations of the training process, the dataset will hence be extended in both the number of included machine states, as well as verbalization variety. Especially the increased variety of prompt / wording combinations may render the two-stage prompting approach unnecessary.

In addition to enhancing the training data, the layout and hyperparameters of the model need additional exploration. The current version of the model is trained with $\alpha = 0.2$, leaving 80 % control to the base model. Also, for the current model a balanced injection strategy was chosen, injecting into the 8th, 16th, 24th, and 30th transformer block of the base model. Other strategies could be to only condition the first few blocks, leaving deeper reasoning untouched. Conversely, only the last blocks could be conditioned exclusively. In addition, the number of latent vectors, as well as the number of attention heads in the adapter can be varied.

In order to reduce computational cost, the relatively small base model `Mistral-7B` was chosen. Using a larger base model could benefit model performance. Larger models have stronger general reasoning, better domain language capabilities, and fewer failure modes. Also, large instruct-aligned models might generalize better from small adapter datasets to unseen machine states and prompt variations.

The most important part of a follow up study, however, will be finding suitable performance metrics. To this end, simulation-based machine states can enable a proper attribution study, as machine parameters can be adjusted freely and independent of real world user operation.

In conclusion, we have shown a first promising version of a machine state embedding model coupled to an open-weights large language model using cross-attention conditioning. Further enhancements to the model and proper attribution studies are expected to bring the model closer to day-to-day operation use-cases, such as anomaly detection, automated shift reporting and general operator assistance.

ACKNOWLEDGMENTS

We acknowledge support from DESY (Hamburg, Germany), a member of the Helmholtz Association HGF and European XFEL GmbH (Schenefeld, Germany). We thank Matthias Scholz (DESY) and Pedro Castro-Garcia (DESY) for their support in creating the training dataset. This research was enabled by the Maxwell computational resources operated at Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Nuclear Physics under contract DE-AC05-06OR23177.

REFERENCES

[1] W. Decking *et al.*, “A MHz-repetition-rate hard X-ray free-electron laser driven by a superconducting linear accelerator,”*Nat. Photonics*, vol. 14, no. 6, pp. 391–397, May 2020. `doi:10.1038/s41566-020-0607-z`

[2] A. Q. Jiang *et al.*, “Mistral 7B”, arXiv:2310.06825 [cs.CL], 2023. `doi:10.48550/arXiv.2310.06825`

[3] A. M. Turner *et al.*, “Steering language models with activation engineering”, arXiv:2308.10248[cs.CL], 2024. `doi:10.48550/arXiv.2308.10248`

[4] J. Wehner *et al.*, “Taxonomy, opportunities, and challenges of representation engineering for large language models”, arXiv:2502.19649 [cs.CL], 2025. `doi:10.48550/arXiv.2502.19649`

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEPD082

[5] H. Zhang, X. Wang, C. Li, X. Ao, and Q. He, "Controlling Large Language Models Through Concept Activation Vectors", in *Proc. AAAI Conf. Artificial Intell.*, vol. 39, no. 24, pp. 25851–25859, Apr. 2025. doi:10.1609/aaai.v39i24.34778

[6] A. Yousefpour *et al.*, "Representation bending for large language model safety", arXiv:2504.01550 [cs.LG], 2025. doi:10.48550/arXiv.2504.01550

[7] GPT-5 system card, https://openai.com/index/gpt-5-system-card/

[8] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library", in *Proc. 33rd Conf. Neural Inf. Process. Systems (NeurIPS 2019)*, Vancouver, Canada, Dec. 2019, pp. 8024–8035. doi:10.48550/arXiv.1912.01703

[9] T. Wolf *et al.*, "HuggingFace's Transformers: State-of-the-art Natural Language Processing", arXiv:1910.03771v5 [cs.CL], Jul. 2020. doi:10.48550/arXiv.1910.03771

[10] DOOCS, https://doocs.desy.de

[11] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE", *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[12] A. Vaswani *et al.*, "Attention is all you need", *Proc. 31st Int. Conf. Neural Inf. Process. Systems (NIPS'17)*, 2017. doi:10.48550/arXiv.1706.03762

[13] A. Grattafiori *et al.*, "The Llama 3 Herd of Models", arXiv:2407.21783v3 [cs.AI], 2024. doi:10.48550/arXiv.2407.21783

[14] mistralai / Mistral-7B-Instruct-v0.3, https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3