University of Nottingham
UK | CHINA | MALAYSIA

Faculty of Computer Science

JupyterLab Extension for Supporting Hyper-parameter Tuning of General Machine Learning Process

INTERIM REPORT

AUTHOR

Jianwen LYU

Student ID: 20320978

scyjl14@nottingham.ac.uk

SUPERVISED BY

Kai Xu

kai.xu@nottingham.ac.uk

Abstract

Though auto-ML approaches makes hyper-parameter tuning require less time and effort, those approaches barely reached on computational notebooks, such as Jupyter. This project is focused on addressing the pervasive challenges of hyper-parameter tuning in general machine learning tasks through a JupyterLab extension. The current progress includes completed preliminary investigations, user requirement collections, and user interface designs. Essential components of the JupyterLab extension framework have been coded, drawing inspiration from the structure of Verdant. User interviews have unveiled critical pain points in the hyper-parameter tuning process, informing the creation of comprehensive user requirement lists. Functional code development is on the horizon, encompassing the implementation of identified functionalities and user interface components. Ongoing user requirement collections and research into alternative hyper-parameter optimization algorithms will drive further refinements. The project reflects an adaptive approach, evolving to address practical limitations and deliver a valuable tool for enhancing the efficiency of hyper-parameter tuning in machine learning workflows.

Contents

1 Introduction

For classical or current machine learning tasks, hyper-parameter tuning, or hyper-parameter optimisation (HPO) is necessary for achieving best accuracy for specific models. This process has been a pain point for data scientists, machine learning model developers, and regular machine learning model users, since hyper-parameter configurations can strongly affect model performance while hyper-parameter tuning requires repetitively sampling from a large number of possible combinations of hyper-parameter values. As dataset sizes and complexity of models are rapidly growing nowadays, the process of hyper-parameter tuning can become longer and more repetitive, intensely increasing the amount of resources and time needed to perform this process (Montanari, Bernardis, and Cremonesi 2022).

In order to address this problem, the concept of Automated Machine Learning (AutoML) is introduced. This technique is aimed for automating the process of machine learning, saving time and resources for manually tuning machine learning model components including hyper-parameters (Yaliang Li et al. 2021). A large number of auto-ML tools can perform automated hyper-parameter tuning and relevant visualizations, making hyper-parameter tuning process much easier, such as HyperTuner (T. Li et al. 2018) or Hyper-Tendril (Park et al. 2021). However, most of these auto-ML tools are built as individual applications or based on other auto-ML platforms, there are few implementations of hyper-parameter tuning supporters on computational notebooks such as Jupyter.

Computational notebooks, as a popular platform for machine learning developments and implementations, have more accessibility to project code and allow for a variety of multi-model data, such as plots and markdown texts (Mcnutt et al. 2023). Taking these advantages, these notebooks can help recognize model structure and obtain part visualizations of model results and performance, making the mode training and tuning process more convenient. However, because there are few supporting tools on this platform at the moment, hyper-parameter tuning process in these notebooks still need repetitive manual works. In this project, a hyper-parameter tuning supporter will be designed and developed to tackle this problem.

2 Motivation

As mentioned in the Introduction, current Jupyter-based or other computational notebook based code assistants rarely implemented automatic hyper-parameter tuning with search space visualizations. These notebooks in fact can be convenient for data analysts or machine learning professionals, by writing interleaved code with individual outputs, which is not supported in traditional code editors, these users can reach an easier parallel code arrangement that requires complex encapsulations in traditional project code (Head et al. 2019). Therefore, it may product more comfortable user experiences for hyper-parameter tuning tools, which have been popularly implemented in traditional code platforms, to be compatible for the computational notebooks as well.

Hence, this project aims to enable hyper-parameter tuning and visualising on computational notebooks. With this project idea, a large number of project structures can be selected, such as individual application that can extract data in computational notebooks and represent in its user interface, or extension that can be directly embedded in the notebooks for more convenience. The hyper-parameter tuning supporter, as the main aim of this project, should support automatic hyper-parameter optimization algorithms and clear visualization of the hyper-parameter search space, helping user to identify the influences of hyper-parameter mutations on model performance. Also, the design of user interface and embedded algorithms should be planned based on user requirements from machine learning majored users of Jupyter Lab, which is a popular computational notebook nowadays. Followed by these considerations, a JupyterLab extension for hyper-parameter tuning support was determined as the product of this project because of the convenience provided for visualizations and the ability to import relevant optimization algorithms directly in python kernels.

3 Related Work

This project is enlightened by previous works of hyper-parameter tuning related machine learning or algorithmic approaches, including hyper-parameter searching techniques and hyper-parameter visualization approaches. However, most of these supporters are developed for traditional code editors or professional auto-ML platforms, while this project will focus on hyper-parameter tuning in computational notebooks.

3.1 Hyper-parameter Optimization

In machine learning, hyper-parameters are often important non-learnable model parameters that can largely affect model performances. Tuning these hyper-parameters to obtain optimal model performances have long been a tricky question. Since manual hyper-parameter tuning tends out to be a tedious process, especially for complex models, several hyper-parameter optimization algorithms were developed in order to tune the hyper-parameters automatically. Regularly used algorithms are including Grid Search, Random Search, Sequential Search, and approaches based on Genetic Algorithms (Bergstra, Bardenet, et al. 2011).

Though the most popular method for machine learning model users is still Grid Search along with searching manually, it turns out that Random Search algorithm comparing to this traditional method can have a larger searching scope of the hyper-parameter space, hence avoid the ignorance of important hyper-parameters. According to the Gaussian process analysis conducted by Bergstra and Bengio, in most circumstances the number of effective hyper-parameters is limited, and this difference of importance among hyper-parameters makes Grid Search suffer from poor coverage in important dimensions, which can be avoided, as the researches stated, Random Search algorithm (Bergstra and Bengio 2012).

Also, evolutionary algorithms for hyper-parameter tuning can be an effective approach in the context of Deep Neural Networks. One example is Multi-node Evolutionary Neural Networks for Deep Learning (MENNDL), an evolutionary algorithm aiming for hyper-parameter optimization in Deep Neural Networks, this method can effectively reduce the need to manually iterate over hyper-parameter search space by determining current search area using past results (Young et al. 2015).

3.2 Visualization of Hyper-parameter Search Space

Except for the tedious hyper-parameter tuning process, another pain point for data analysts is that they cannot have a clear view of the hyper-parameter search space. To tackle this problem, they need to record their hyper-parameter settings and corresponding model performances by hand, taking much time and effort in this and the following analysis process. This problem still occurs after automatic hyper-parameter tuning algorithms are used, for data analysts often lack specific knowledge of these algorithms, hence cannot figure out how the hyper-parameters will be modified by the algorithm they use.

Several visualization tools are developed in order to tackle this problem. HyperTendril by Park et al. provides high-level visualization based on auto machine learning platforms which enables user relevant plots of hyper-parameter search space as well as a diagnostic view of hyper-parameter optimization algorithm components (Park et al. 2021). Also, HyperTuner by Li et al. provides visual analytic for hyper-parameter tuning, which can achieve human-in-the-loop approach of hyper-parameter tuning and satisfy the needs of relevant search space visualizations (T. Li et al. 2018). As a more advanced approach, Li et al. integrated asynchronous scheduling and multi-fidelity optimizer into hyper-parameter optimization process in their work HyperTune, making the system more efficient and robust (Yang Li et al. 2022).

3.3 Coding-assistants in Computational Notebooks

While prior literature touches upon a limited number of machine learning and hyper-parameter tuning supporters within computational notebooks like JupyterLab, the current landscape is rich with a plethora of code assistants and visualization tools catered to machine learning and programming professionals. A comprehensive exploration of these existing works promises valuable insights into design patterns and user interface presentation techniques that can greatly inform the development of the current project.

Among the notable code assistants, Mage stands out as a computational notebook API designed to facilitate seamless transitions between a user's code and graphical works, thereby enhancing the flexibility of tool selections for various computing tasks (Kery, Ren, et al. 2020). In the realm of data visualization, the B2 library serves as an illustrative example, empowering users to bridge their data frames with corresponding visualizations within computational notebooks, thereby facilitating interactive data analysis (Wu, Hellerstein, and Satyanarayan 2020). Additionally, NBSearch offers an instructive perspective by allowing users to perform semantic-based code searches in computational notebooks. The embedded search engine utilizes machine learning techniques, rendering it adaptable to natural language search queries and providing intuitive visualizations (X. Li et al. 2021), which provides considerable insights into the user interaction functionalities for this project. By delving into these diverse tools and approaches, the project gains a broader understanding of successful implementations in computational notebooks, aiding in the refinement of its own design and functionality.

4 Description of the Work

4.1 Methodology

This project can be divided into user requirement collecting and developing sections. In the user requirement collecting section, several user interviews for machine learning model users were conducted in order to identify main difficulties in the hyper-parameter tuning process. In the development part, a JupyterLab extension will be developed for hyper-parameter tuning support, JavaScript frameworks will be implemented such as React.js or D3.js in order to present clear and intuitive user interfaces and model performance visualizations.

The user requirements were collected mainly by interviews with participants, in this project, computer and data analysis professionals (who were also JupyterLab users) were invited to a short interview with questions related to their hyper-parameter tuning experiences, and difficulties they met during this process. Also, advice from these professionals for hyper-parameter and model performance visualizations were documented, the professionals were asked for their preferred forms of visualization, which can provide ideas for the interface design in development section, which will be based on the requirement lists concluded from these interviews.

After the development process, the prototypes will be evaluated by the participants, and further refinements will be made based on their feedback. This feedback collection will be in form of questionnaires with questions related to user experiences, improvements comparing to traditional parameter tuning methods, and potential problems. The user feedback and refinement process will be iterative until user feel satisfied about the product.

4.2 Expected Functionalities of Product

As a hyper-parameter tuning supporter in general machine learning process on JupyterLab. The extension should satisfy the functionalities of 1) automatic hyper-parameter tuning, and 2) visualization of hyper-parameter search space along with model performance.

4.2.1 Automatic Hyper-parameter Tuning

Though the concept of autoML brings much convenience to hyper-parameter optimization process, this process often varies since the model type, model complexity and data distribution may be different for machine learning tasks. In the context of model building or data analysing on computational notebooks, the user usually needs prior works of identifying model to use and during this process, hyper-parameters should be filled into the corresponding interface for user to check, as the first step of automatic hyper-parameter tuning.

After all the hyper-parameters are determined by the system, the second step of automatic hyper-parameter tuning procedure should be achieved by the embedded hyper-parameter optimization algorithms in the extension, including traditional Grid Search, Random Search and evolutionary algorithm for hyper-parameter optimization. Via the interface of extension, user should be able to select which algorithm to apply depending on their model complexity and actual needs and run the iterative testing section provided by the extension. In the back-end part of the product, user specified number of validation rounds should be performed and model performance of each round, along with the record of mutations on hyper-parameter values will be stored for further analysis.

4.2.2 Visualization of Hyper-parameter Search Space

As mentioned by previous works of hyper-parameter visualizations for regular code editors, the main components of visualization should be focused on model performance and trends of hyper-parameter changes during the testing process. Since these visualizations should help users quickly identify influences of each hyper-parameter on model performance, and which parameter may contribute more to the variation of model performance, the visualization part of the project should combine these views together for better recognition.

After these considerations, the visualization part should have a user view in form of plots of model performance with respect to the change of each hyper-parameter, for obtaining a clear view of the influences of hyper-parameter mutations on model performance. Also, a ranking functionality should be satisfied for user to identify which hyper-parameter has the most profound influence on model performance. With these functionalities in the extension, the hyper-parameter tuning step of general machine learning on computational notebooks will probably be more convenient and intuitive for users.

5 Design

5.1 Principles for Project UI Design

Through the design space study of (Mcnutt et al. 2023) by McNutt et al., several worth-noting points while designing user interfaces of computational notebook assistants were presented. Although the focus of this study was code assistants such as GitHub Copilot, these findings can still provide some enlightenment for UI design part of this project. Based on these enlightenment several design requirements are established, including 1) Connection with User Code 2) Data Provenance and 3) Customization.

5.1.1 Connection with User Code

The user interface is meticulously crafted to establish an interactive and seamless connection with the user's project code. When engaged in machine learning tasks utilizing the extension, the user interface will dynamically showcase all relevant hyper-parameters associated with the user's preferred model. Importantly, it incorporates a sophisticated functionality that enables users to not only view but also modify these hyper-parameter values directly within the user interface, fostering a harmonious interaction between the extension and the user's underlying project code.

This bidirectional interaction ensures that any adjustments made to hyper-parameters in the user interface are mirrored in the project code, and vice versa. This symbiotic relationship streamlines the user experience, offering a level of convenience that empowers users to effortlessly fine-tune hyper-parameter settings without the need for navigating through intricate code structures. By providing this dual-view functionality, the user interface will facilitate a more intuitive and user-friendly environment for managing hyper-parameters, also making the design choice aligns with the overarching goal of enhancing user convenience and efficiency throughout the machine learning workflow.

5.1.2 Data Provenance

The validation process of this hyper-parameter tuning extension is meticulously crafted to cater to the principles of data provenance and retrieval. Upon the completion of a specified number of testing rounds, users gain access to a comprehensive overview of hyper-parameter settings, both current and previous, conveniently stored within the history box. Within this repository of historical configurations, users are afforded the capability to seamlessly replicate previous settings into their ongoing experiments, fostering a sense of continuity and informed decision-making.

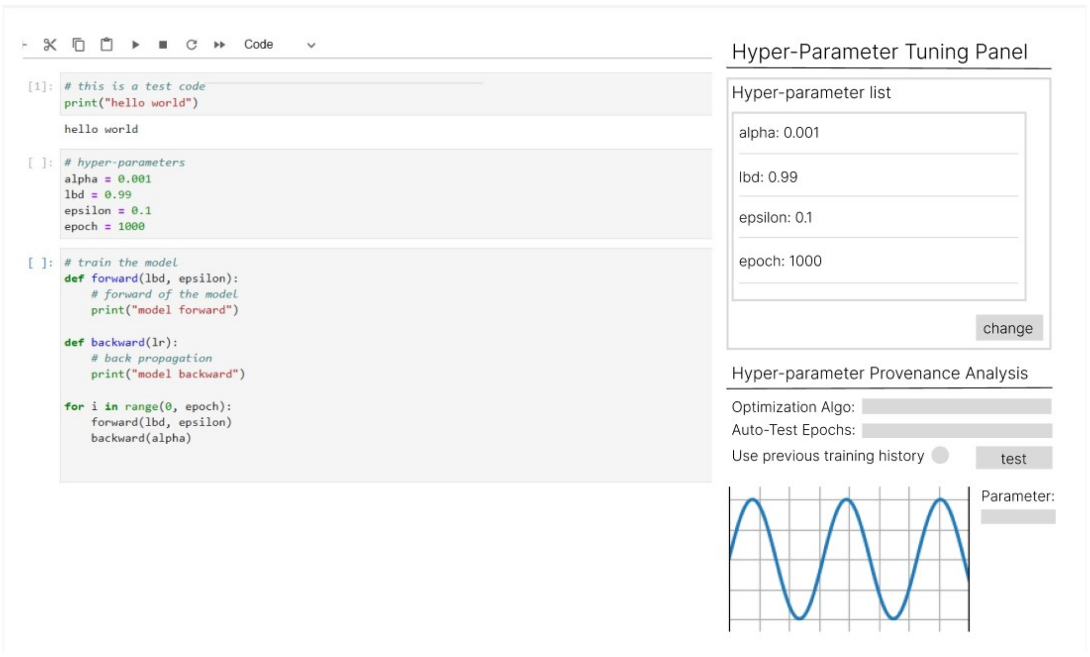
In addition to traditional testing methods, when employing evolutionary algorithms for hyper-parameter tuning, users are enabled with the choice to check previous search histories. This approach to data provenance serves as an additional layer of insight, allowing users to make informed decisions by drawing upon the information encapsulated in their historical experiments. These data provenance mechanisms play an important role in offering ability to modify configurations for subsequent rounds based on insights gleaned from past experiences. This holistic approach underscores the commitment to providing users with a robust and informed environment for optimizing their machine learning models.

5.1.3 Customization

The customization aspect will be thoroughly addressed in the design of the extension, ensuring a user-centered experience. A multitude of user-configurable parameters will be embedded within the hyper-parameter tuning workflow facilitated by the extension. This encompasses empowering users with the flexibility to effortlessly modify hyper-parameter values directly through the user interface. Additionally, users will have the discretion to meticulously choose from a range of optimization algorithms tailored to their specific hyper-parameter tuning needs.

A pivotal element of the customization features is the provision for users to seamlessly integrate their own hyper-parameter configurations into the system. This intentional design choice aims to foster a sense of ownership and adaptability, enabling users to tailor the extension to their unique preferences and requirements. Such customization avenues play a crucial role in expediting the learning curve, allowing users to swiftly acclimate to the hyper-parameter tuning procedures facilitated by the extension's intuitive assistant. The emphasis on user empowerment and adaptability underscores the commitment to providing a tailored and user-friendly experience.

5.2 User Interface Prototype



```
[1]: # this is a test code
     print("hello world")

     hello world

[ ]: # hyper-parameters
     alpha = 0.001
     lbd = 0.99
     epsilon = 0.1
     epoch = 1000

[ ]: # train the model
     def forward(lbd, epsilon):
         # forward of the model
         print("model forward")

     def backward(lr):
         # back propagation
         print("model backward")

     for i in range(0, epoch):
         forward(lbd, epsilon)
         backward(alpha)
```

Hyper-parameter list

alpha: 0.001

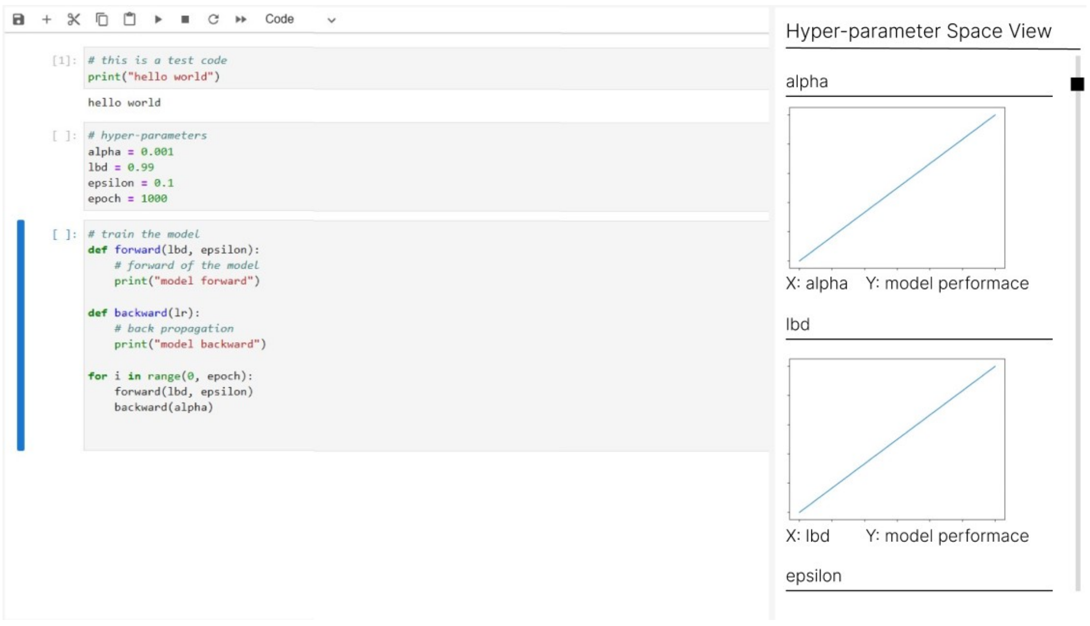
lbd: 0.99

epsilon: 0.1

epoch: 1000

Code
Hyper-Parameter Tuning Panel
change
Hyper-parameter Provenance Analysis
Optimization Algo:
Auto-Test Epochs:
Use previous training history
test
Parameter:

Figure 1: hyper-parameter tuning panel



```
[1]: # this is a test code
     print("hello world")

     hello world

[ ]: # hyper-parameters
     alpha = 0.001
     lbd = 0.99
     epsilon = 0.1
     epoch = 1000

[ ]: # train the model
     def forward(lbd, epsilon):
         # forward of the model
         print("model forward")

     def backward(lr):
         # back propagation
         print("model backward")

     for i in range(0, epoch):
         forward(lbd, epsilon)
         backward(alpha)
```

Code
Hyper-parameter Space View
alpha
X: alpha Y: model performace
lbd
X: lbd Y: model performace
epsilon

Figure 2: hyper-parameter plots view

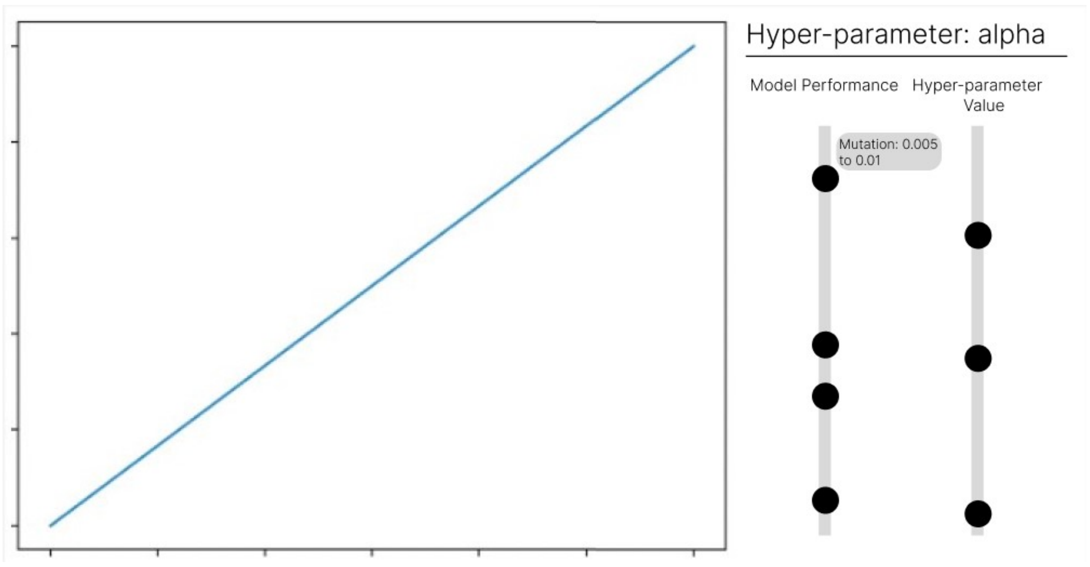
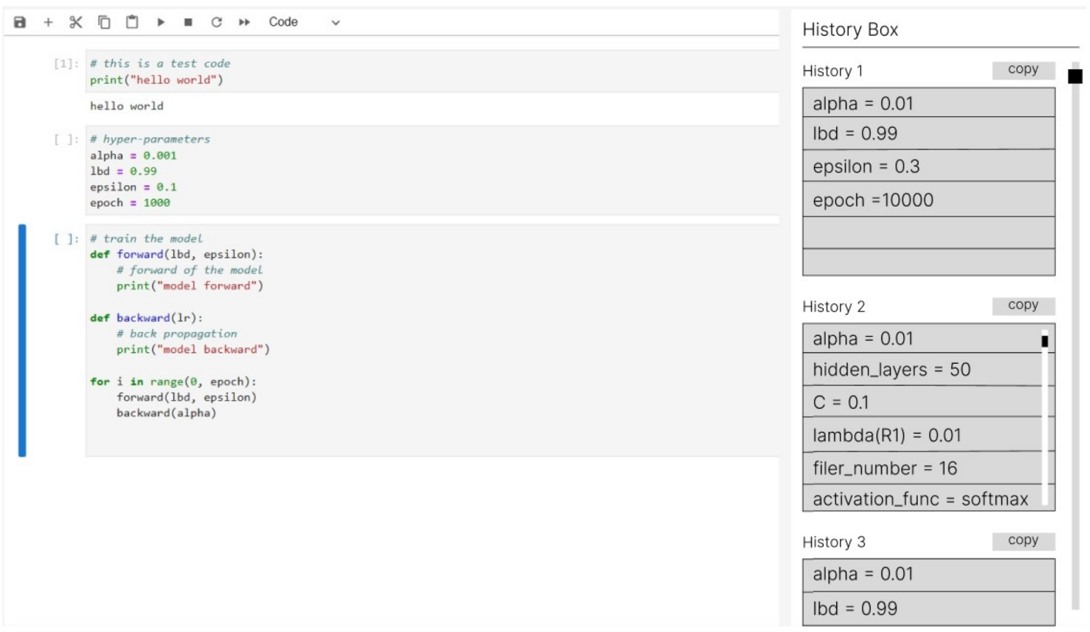


Figure 3: hyper-parameter plots view



```
[1]: # this is a test code
     print("hello world")

     hello world

[ ]: # hyper-parameters
     alpha = 0.001
     lbd = 0.99
     epsilon = 0.1
     epoch = 1000

[ ]: # train the model
     def forward(lbd, epsilon):
         # forward of the model
         print("model forward")

     def backward(lr):
         # back propagation
         print("model backward")

     for i in range(0, epoch):
         forward(lbd, epsilon)
         backward(alpha)
```

History 1

```
alpha = 0.01
lbd = 0.99
epsilon = 0.3
epoch =10000
```

History 2

```
alpha = 0.01
hidden_layers = 50
C = 0.1
lambda(R1) = 0.01
filer_number = 16
activation_func = softmax
```

History 3

```
alpha = 0.01
lbd = 0.99
```

Figure 4: hyper-parameter history box

5.2.1 Overview Panel

As illustrated in Figure 1, the layout and functionality of the hyper-parameter tuning control panel are described. The control panel is designed to be a window seamlessly integrated with the user's project code, positioned adjacent to the code cells. The interface components consist of a list view displaying the current hyper-parameters. In this section, users are required to make sure that the list is filled with the hyper-parameters relevant to their model. Furthermore, users can dynamically modify the hyper-parameter values directly in the code by making corresponding changes in the list view. Additionally, a hyper-parameter analysis panel is incorporated to visualize the hyper-parameter search space. Figure 1 presents a singular view depicting the model performance in relation to a specific hyper-parameter. Users can choose test epochs for the optimization algorithm, decide whether to consider historical trails, and select which hyper-parameter influences they want to visualize in the plot. In the actual development environment, the interface will also encompass an integrated view illustrating the impact of each hyper-parameter on the model performance, which will be explained in next sub-section.

5.2.2 Split View of Plots

It is obvious that the user cannot easily obtain full control and knowledge of hyper-parameter space, therefore, the user interface is designed to have split-views of each hyper-parameter, as shown in Figure 2. In this part of the extension user interface, all hyper-parameter listed in the hyper-parameter list view will be presented with their influences on model performance. These influences will be shown as plots with x-axis as hyper-parameter values, and y-axis as model-performance score. Hyper-parameters can be identified as influential when their changes triggers followed changes on model performance or model converge state (such as oscillations). Also, user can identify which hyper-parameter is more important for shaping model performance by observing the extent of influences each parameter has.

While this split-view may still be confusing for users to understand the hyper-parameter tuning process, a detailed view of each plot will be emerged when user click the plot, as shown in Figure 3. In this detailed view, the mutations of hyper-parameters and changes in model performance will be recorded in a chart, indicating at which time these changes occurred. By observing this detailed view, the user will be able to figure out if some changes in a specific hyper-parameter value triggered a change in the model performance, which probably indicates that this parameter has rather important influences to the results of the machine learning model. Moreover, These changing points are designed to be interactive in the visualization interface, when user click these points, a tip will inform user the explicit information about which hyper-parameter changed for what extent, or how many the model performance changed. By checking these information, users can also compare which parameter triggered more changes in model performance, and have a impression of which parameter tends to be more important than others.

5.2.3 History Box

The user interface also incorporates a history box to meticulously archive past records, illustrated in Figure 4. This facet of the design aims to afford users a more comprehensive data provenance, facilitating the retrieval of prior hyper-parameter configurations. This feature enhances the user's adaptability in the hyper-parameter tuning process, offering the ability to revert to previous configurations should the current results prove unsatisfactory, allowing for the exploration of new optimization directions from a prior state. The historical configurations of hyper-parameters are systematically stored in a list view, enabling users to seamlessly replicate these values for their ongoing optimization processes. It's imperative to note that if a user's current configurations comprise a different number of hyper-parameters than their historical records, this replication process becomes unattainable, potentially signaling a shift to a different model by the user.

6 Progress

6.1 Topic Alternations in Previous Developing Process

This project, influenced by a large number of limitations and realistic considerations, have been through a convoluted process of finding proper users and developing project topic. There were topic alternations during the project after communications with the sponsor, triggered by factors such as hardware and user type limits.

In the initial stage, the original project topic is about generative Artificial Intelligence for Artists, this topic attracted the researcher by its novelty and high relevance to the field of computer graphics. At first the project idea is to develop a Jupyter extension for Disco Diffusion, a large image generative model, which can achieve automatic recommendations of keywords in user prompt. This idea is then negated since the GPU configuration, especially video memory, in the local device of researcher cannot support these large models. Only small images such as 100x100 can be generated, while these generated images lack details and at most time only presented as a slightly processed noise image with random colors. Servers or Jupyer-based online coding platforms such as Google Colab was then considered, but these platforms cannot support Jupyter extensions.

After this approach, the project topic changed to developing a web browser extension for Dream Studio, an online image generating platform. However, though the development of this extension can avoid limitations of GPU in the local device, one problem still cannot be addressed: the potential users of this project were required to satisfy two conditions. First, they have to be artists or people interested and have experiences in art-related activities such as animations or comics. Second, they have to be users of generative image models, since the researcher have to collect user experiences of these models in order to design the user prompt refinement functionality. After investigations, there seemed to have very few qualified users for the interviews, which means the user requirements may be not enough to bring insights in to the extension design. Driven by considerations based on this limitation, the project topic was forced to have another alternation.

After communications with project sponsor, the project topic was permitted to shift to general machine learning tasks instead of generative image models. Given by this new topic, the hyper-parameter tuning process of machine learning workflow was chosen as the focus of the project, since many surrounding people suffer from this monotonous iterative process, which means there will be enough potential users. Also, general machine learning process has less requirements for model size and complexity during the testing process, which can avoid hardware limitations during the development process. Hence, the project topic transferred to JupyterLab extension for hyper-parameter tuning in general machine learning process, which is the current project topic.

6.2 Current Progress

Currently the project has completed several preliminary investigations, user requirement collections, and user interface designs. Part of the project code, including the essential Jupyter extension components and frameworks have completed as well.

6.2.1 User Requirements

Through insightful user interviews, a profound understanding of the challenges embedded within the hyper-parameter tuning process has surfaced, shedding light on pivotal pain points that warrant meticulous consideration for the success of the project.

One of the interviewees, for instance, eloquently articulated the struggle associated with comprehending the intricacies of the hyper-parameter space. Specifically, he grappled with the complexity inherent in his style-transfer model, emphasizing the need for an interface that renders the parameter space more intuitive and comprehensible. This valuable user experience underscores the importance of enhancing the user interface's capacity to convey the nuances of hyper-parameters during the tuning process.

Another user, reflecting on his experience, brought attention to the drawbacks of employing the traditional Grid Search method. The formidable time and computational power requirements associated with this approach emerged as significant pain points in his project. Moreover, he highlighted the method's limited ability to thoroughly explore the expansive search space, which, in turn, posed the risk of over-fitting. This user's firsthand account illuminates the critical need for efficient and resource-conscious hyper-parameter optimization algorithms that circumvents the pitfalls of exhaustive search methods.

These invaluable insights from user interviews have been integrated into a comprehensive list of User Requirements, laying the foundation for a user-centered hyper-parameter tuning extension. Foremost among these requirements is the imperative to enhance the user interface's interpretability of hyper-parameter spaces: Most users emphasized the need for an intuitive representation that demystifies the complexities of these spaces, particularly in complex models such as Neural Networks. Additionally, a key requirement emerged while many users underscored the necessity for hyper-parameter optimization strategies that can mitigate the formidable time and computational demands of exhaustive methods, minimizing the risk of over-fitting. Moreover, many users expressed a desire for a more interactive interface that fosters seamless communication between the extension and project code, enabling dynamic modifications and providing a harmonious user experience. These requirements collectively inform the development process, guiding the creation of a solution that not only addresses identified pain points but also aligns with the diverse needs and expectations of practitioners engaged in the hyper-parameter tuning intricacies of machine learning workflows.

6.2.2 Extension Development

The intricate development process of the JupyterLab extension unfolds with the construction of a robust framework utilizing Jupyter kernel operation related APIs in JavaScript. These APIs serve as instrumental tools empowering developers to seamlessly interface with Python kernels within the Jupyter platform, which enables the extraction of data and the identification of specific code components from the user's project code, laying a foundational groundwork for the extension's core operations. Drawing inspiration from the successful framework employed in Verdant, a JupyterLab extension dedicated to aiding data analysts in retracing their data exploration steps (Kery, John, et al. 2019), the project strategically incorporates proven UI design patterns and resource hierarchies. While the specific objectives of Verdant may differ from the current project's focus, the shared emphasis on enhancing user interactions and facilitating a more intuitive workflow provides valuable insights. Leveraging the enlightenment from Verdant's design choices, the researcher is poised to optimize the user interface, ensuring it aligns seamlessly with the JupyterLab environment and caters to the distinctive requirements of users engaged in hyper-parameter tuning for machine learning models. This cross-pollination of design principles positions the project for success in delivering a user-friendly and effective JupyterLab extension for hyper-parameter tuning.

6.3 Future Plan

As the project advances, the researcher will embark on the implementation of functional code, encompassing the development of code to realize the functionalities outlined in earlier sections, in addition to the user interface. Further refinement of the designed prototype will be facilitated through the collection of additional user requirements. Simultaneously, an extensive exploration of related works pertaining to automatic hyper-parameter tuning algorithms will be undertaken. This comprehensive analysis aims to identify and analyze various alternative optimization algorithms that users can potentially select for their specific needs. This iterative process ensures that the final implementation is not only robust and functional but also aligns closely with the preferences and requirements of the target users.

In addition, the exploration of extra functionalities that seamlessly integrate with the existing features of this extension is underway. For instance, on the basis of existing extension structure, there is a consideration for incorporating a model selection functionality based on the user's coding history. This addition would contribute to a more smooth machine learning process within the computational notebook, enhancing the ease of control in hyper-parameter tuning endeavors. Extra article reading, user feedback collection and coding works will be launched once these additional functionalities are confirmed as useful to user's model building and optimizations.

References

Bergstra, James, Rémi Bardenet, et al. (2011). "Algorithms for Hyper-Parameter Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc.

Bergstra, James and Yoshua Bengio (Feb. 2012). "Random Search for Hyper-Parameter Optimization". In: *J. Mach. Learn. Res.* 13.null, pp. 281–305. ISSN: 1532-4435.

Head, Andrew et al. (2019). "Managing Messes in Computational Notebooks". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. Glasgow, Scotland Uk: Association for Computing Machinery, pp. 1–12. ISBN: 9781450359702. DOI: 10.1145/3290605.3300500. URL: https://doi.org/10.1145/3290605.3300500.

Kery, Mary Beth, Bonnie E. John, et al. (2019). "Towards Effective Foraging by Data Scientists to Find Past Analysis Choices". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. Glasgow, Scotland Uk: Association for Computing Machinery, pp. 1–13. ISBN: 9781450359702. DOI: 10.1145/3290605.3300322. URL: https://doi.org/10.1145/3290605.3300322.

Kery, Mary Beth, Donghao Ren, et al. (2020). "Mage: Fluid Moves Between Code and Graphical Work in Computational Notebooks". In: *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. UIST '20. Virtual Event, USA: Association for Computing Machinery, pp. 140–151. ISBN: 9781450375146. DOI: 10.1145/3379337.3415842. URL: https://doi.org/10.1145/3379337.3415842.

Li, Tianyi et al. (2018). "Hypertuner: Visual analytics for hyperparameter tuning by professionals". In: *2018 IEEE Workshop on Machine Learning from User Interaction for Visualization and Analytics (MLUI)*. DOI: 10.1109/mlui52768.2018.10075647.

Li, Xingjun et al. (2021). "NBSearch: Semantic Search and Visual Exploration of Computational Notebooks". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. <conf-loc>, <city>Yokohama</city>, <country>Japan</country>, </conf-loc>: Association for Computing Machinery. ISBN: 9781450380966. DOI: 10.1145/3411764.3445048. URL: https://doi.org/10.1145/3411764.3445048.

Li, Yaliang et al. (2021). "AutoML: A Perspective Where Industry Meets Academy". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. KDD '21. Virtual Event, Singapore: Association for Computing Machinery, pp. 4048–4049. ISBN: 9781450383325. DOI: 10.1145/3447548.3470827. URL: https://doi.org/10.1145/3447548.3470827.

Li, Yang et al. (Feb. 2022). "Hyper-Tune: Towards Efficient Hyper-Parameter Tuning at Scale". In: *Proc. VLDB Endow.* 15.6, pp. 1256–1265. ISSN: 2150-8097. DOI: 10.14778/3514061.3514071. URL: https://doi.org/10.14778/3514061.3514071.

Mcnutt, Andrew M et al. (2023). "On the Design of AI-Powered Code Assistants for Notebooks". In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. CHI '23. New York, NY, USA: Association for Computing Machinery. ISBN: 9781450394215. DOI: 10.1145/3544548.3580940. URL: https://doi.org/10.1145/3544548.3580940.

Montanari, Matteo, Cesare Bernardis, and Paolo Cremonesi (2022). "On the Impact of Data Sampling on Hyper-Parameter Optimisation of Recommendation Algorithms". In: *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. SAC '22. Virtual Event: Association for Computing Machinery, pp. 1399–1402. ISBN: 9781450387132. DOI: 10.1145/3477314.3507158. URL: https://doi.org/10.1145/3477314.3507158.

Park, Heungseok et al. (2021). "HyperTendril: Visual analytics for user-driven hyperparameter optimization of deep neural networks". In: *IEEE Transactions on Visualization and Computer Graphics* 27.2, pp. 1407–1416. DOI: 10.1109/tvcg.2020.3030380.

Wu, Yifan, Joseph M. Hellerstein, and Arvind Satyanarayan (2020). "B2: Bridging Code and Interactive Visualization in Computational Notebooks". In: *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. UIST '20. Virtual Event, USA: Association for Computing Machinery, pp. 152–165. ISBN: 9781450375146. DOI: 10.1145/3379337.3415851. URL: https://doi.org/10.1145/3379337.3415851.

Young, Steven R. et al. (2015). "Optimizing Deep Learning Hyper-Parameters through an Evolutionary Algorithm". In: *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*. MLHPC '15. Austin, Texas: Association for Computing Machinery. ISBN: 9781450340069.

DOI: `10.1145/2834892.2834896`. URL: `https://doi.org/10.1145/2834892.2834896`.