CBM-GUI: GUI for Concept Bottleneck Layer Integration in LLMs

Gabriel Cha
gcha@ucsd.edu

Steven Luong
sxluong@ucsd.edu

Tsui-Wei (Lily) Weng
lweng@ucsd.edu

Abstract

Large Language Models (LLMs) have shown remarkable capabilities, but their ”black-box” nature often makes them difficult to interpret. Concept Bottleneck Layers (CBLs) address this by linking model outputs to human-understandable concepts. Our project introduces a user-friendly Graphical User Interface (GUI) that allows users, regardless of their technical background, to easily integrate CBLs into pre-trained LLMs (provided as.pt files), display top N highly activated concepts, and ability to analyze and prune biased concepts. By identifying and unlearning biased concepts, these models can produce fairer, more reliable predictions. Our end goal is to provide users with the ability to scrutinize and adjust models to be explainable and fair. This work builds upon the research and framework developed by Weng et al. (2024).

Code: `https://github.com/gabrielchasukjin/ConceptBottleneck-GUI-Experiment/tree/main`

1 Background and Related Work

Large Language Models (LLMs) excel in various tasks but often lack transparency. Concept bottleneck layers (CBLs) allow for transparency by connecting model outputs to human-readable concepts. However, existing CBL implementations require technical expertise, restricting their accessibility. Our project builds on Weng et al. (2024) to create an easy-to-use GUI, allowing users of all skill levels to visualize concept activations, adjust weights, and unlearn biased concepts for greater interpretability, trustworthiness, and fairness.

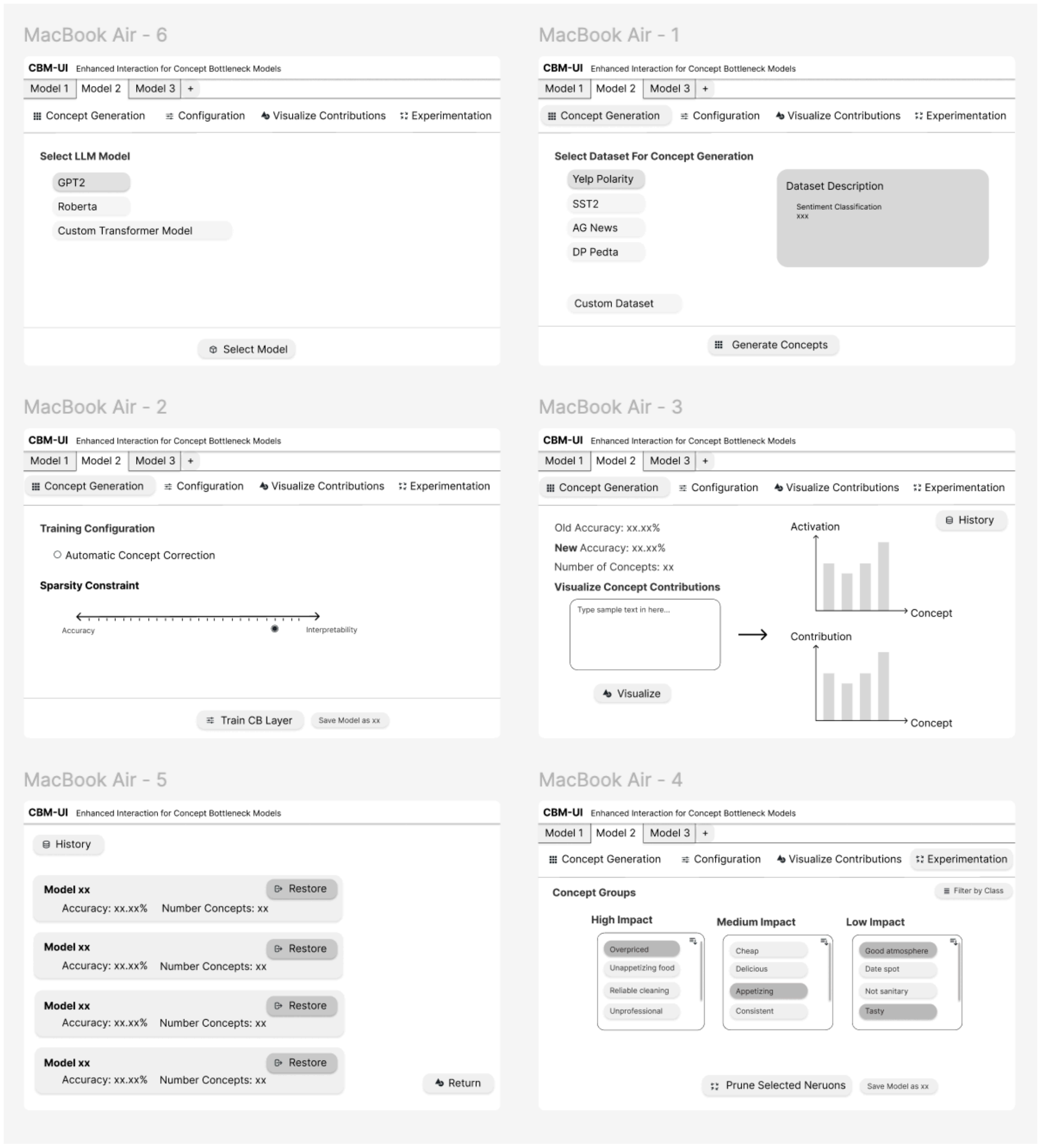

Figure 1: CBM-GUI Workflow: Simplified Interface for Model Transparency and Fairness

2 Minimal Viable Product

The user interface depicted in Figure 1 represents our ultimate vision and serves as the primary inspiration for this project. Our aim is to create an intuitive and accessible platform that enables users to seamlessly integrate Concept Bottleneck Layers (CBLs) into Large Language Models (LLMs). This interface embodies the goal of fostering transparency, fairness, and trust in AI systems.

Currently, we are focused on developing a minimal viable product (MVP) that captures the core functionality of this vision. The MVP will provide essential features such as model configuration, concept visualization, and bias mitigation. This iterative approach ensures we can refine the tool and ultimately converge toward the comprehensive interface shown in Figure 1.

2.1 Home Page (MVP)

The home page of the CBM-GUI Minimum Viable Product (MVP) allows users to select hardware, train the Concept Bottleneck Layer (CBL), and classify inputs.

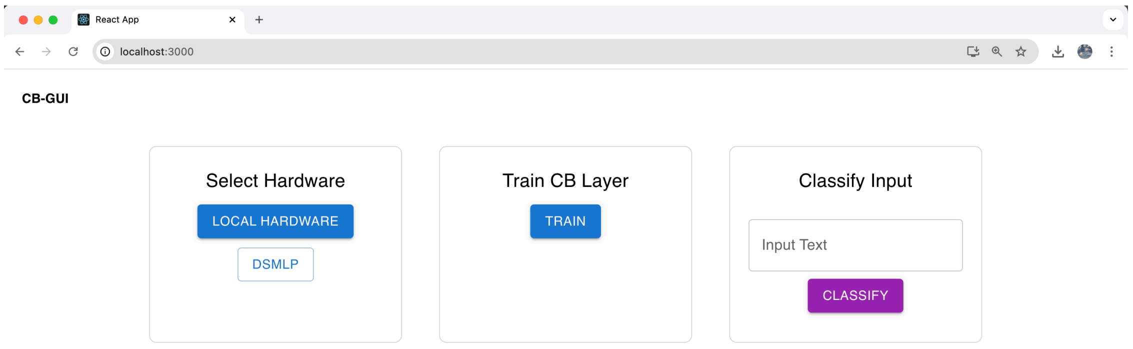

Figure 2: CBM-GUI MVP Homepage

Users can choose to train the CBL layer on either local hardware or the DSMPL UCSD server. Once the hardware is selected, users can click ”Train” to start the training process. Training duration varies based on the hardware, typically taking between 5 to 45 minutes.

For the MVP, the model is pre-selected as a fine-tuned RoBERTa model from Google, and the CBL layer is trained on the SetFit SST2 dataset, which includes 6,920 movie review samples labeled as positive or negative. After training, users can input text for classification and view the top concept contributions that influenced the model's decision-making process.

2.2 Concept Contributions (MVP)

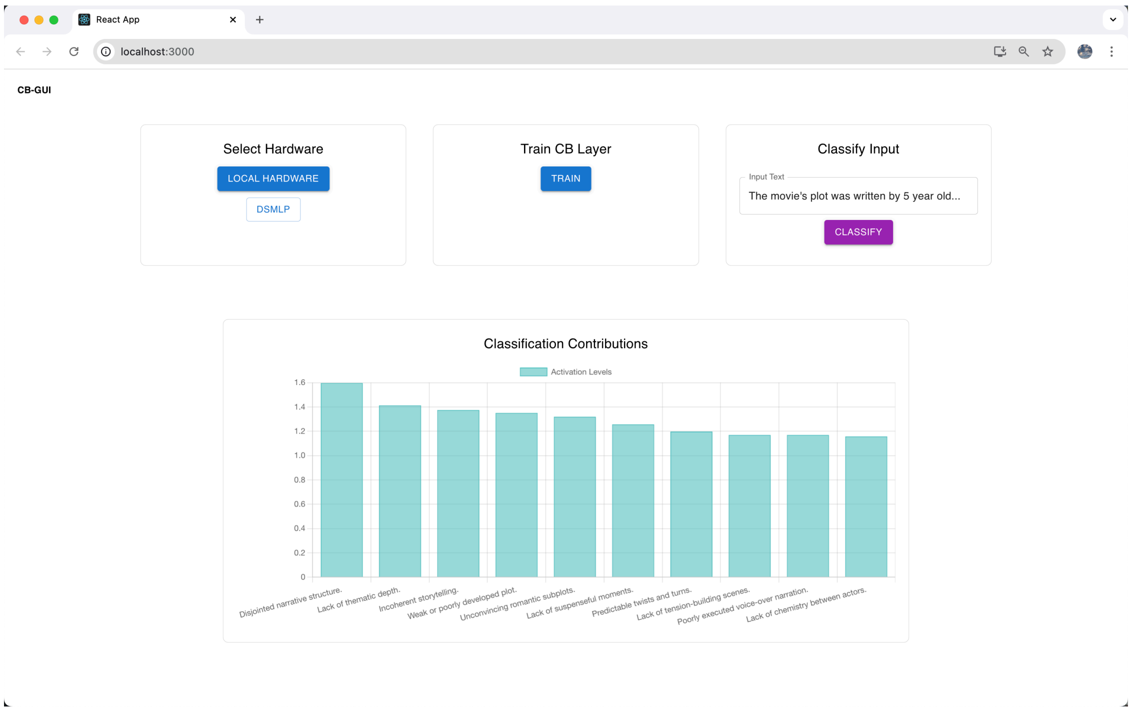
React App
localhost:3000
CB-GUI
Select Hardware
LOCAL HARDWARE
DSMLP
Train CB Layer
TRAIN
Classify Input
Input Text
The movie's plot was written by 5 year old...
CLASSIFY
Classification Contributions
Activation Levels
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
Disjointed narrative structure.
Lack of thematic depth.
Incoherent storytelling.
Weak or poorly developed plot.
Unconvincing romantic subplots.
Lack of suspenseful moments.
Predictable twists and turns.
Lack of tension-building scenes.
Poorly executed voice-over narration.
Lack of chemistry between actors.

Figure 3: Top concept contributions for the input ”The plot was written by a 5-year-old.”

Figure 3 demonstrates the CBM-GUI's ability to classify text and highlight the top concept contributions influencing the model's decision-making process. For the input ”The plot was written by a 5-year-old,” the model identifies key concepts such as ”Disjointed narrative structure,” ”Lack of thematic depth,” ”Incoherent storytelling,” and ”Weak or poorly developed plot” as the primary drivers behind the classification. These contributions are visually represented in a bar chart, where the activation levels of each concept indicate their relative importance. This functionality provides users with a clear and interpretable view of how the model reaches its conclusions.

3 Tech Stack

The proposed GUI will use React for the frontend and Flask (Python) for the backend. React will handle the dynamic UI and state management, while Flask will manage API requests and interactions with the model training microservice. When a user interacts with the UI, React will send a request to the Flask backend, triggering model training or retrieving results. The backend will process the request, store necessary data, and send the response back to React, which will update the UI with real-time progress and results.

4 Features and Method

The MVP shown in Figures 2 and 3 represents the foundational version of our CBM-GUI, demonstrating core functionalities such as hardware selection, CBL training, text classification, and concept contribution visualization. However, this is only the starting point. The following sections outline the new features and methods we plan to implement in the coming months to create a more comprehensive and seamless user experience. These enhancements aim to expand functionality, improve usability, and bring us closer to our ultimate goal of making model interpretability and fairness accessible and practical for all users.

4.1 Dataset Selection

Dataset selection is the first step in the workflow, as it provides the foundation for training the CBL layer. The CBL layer requires data to learn from, using the selected dataset to identify and generate human-understandable concepts that explain the model's behavior.

Users can choose from preloaded datasets (e.g., Yelp Polarity, SST2) or upload their own. A description panel accompanies each option, helping users understand the purpose and relevance of the datasets.

User Flow: After selecting a dataset, users proceed by clicking ”Generate Concepts,” which initiates the backend process of training the CBL layer and extracting relevant concepts linked to the dataset. See Figure 6 in the appendix.

4.2 Model Configuration

Users can adjust the model by toggling automatic concept correction and setting sparsity levels with a slider to balance accuracy and interpretability. **User Flow**: After setting the desired configurations, users click ”Train CB Layer” to initiate the model training process with the configured settings. See Figure 7 in the appendix.

4.3 Visualize Contributions

Users input text to see how concepts activate and contribute to predictions. The interface shows accuracy of the model and the number of active concepts. This visualization helps users understand which concepts are driving the model's predictions and facilitates transparency in decision-making.

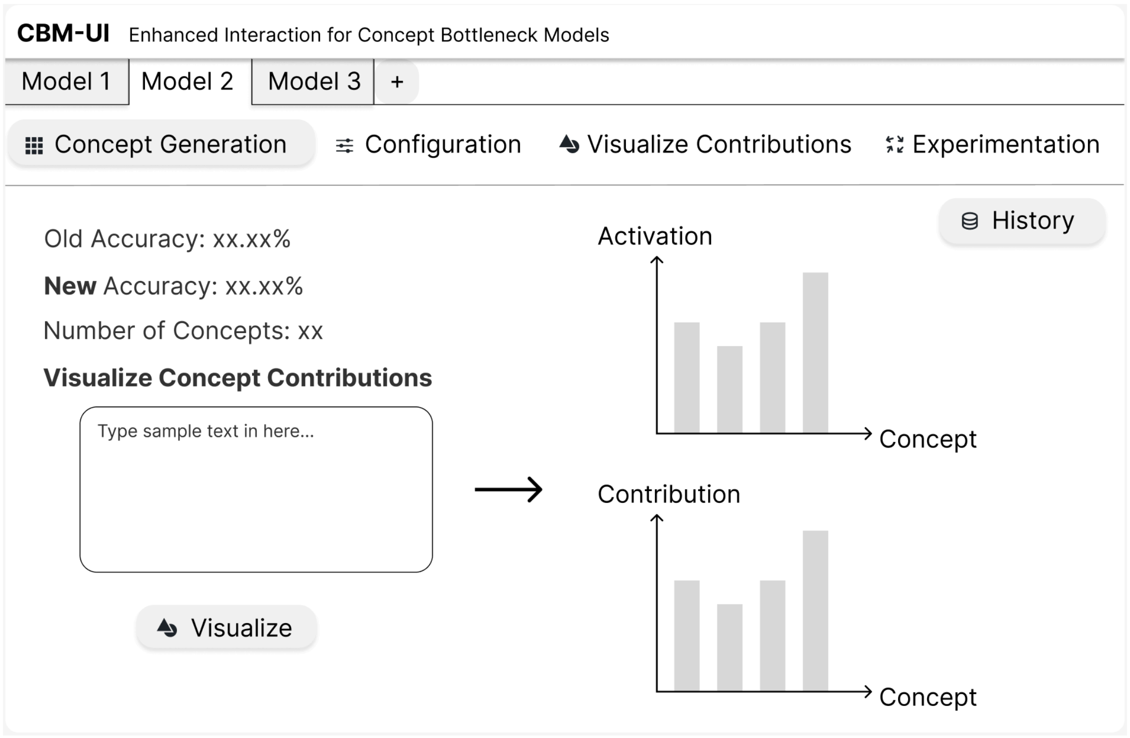

Figure 4: Users can compare how different configurations affect the model's decision-making process by interacting with the visualization tools.

4.4 Experimentation and Pruning

Users can prune concepts grouped by impact (High, Medium, Low) to reduce bias or remove less important ones. After selecting concepts, they re-train the model and see updated accuracy and classification changes. This helps improve fairness and robustness. Users can save the modified model for future use. **User Flow**: After selecting concepts to prune, users can save the modified model for further use. Additionally, users can filter based on classification and sort based on highest contributions. See Figure 8 in the appendix.

4.5 History and Restoration

The history tab shows past models with their accuracy and concept counts. Users can restore any model by clicking "Restore." This allows easy comparison and supports iterative model improvements and reproducibility. **User Flow**: Users can revisit and compare different configurations, facilitating iterative experimentation. See Figure 9 in the appendix.

5 Conclusion

The proposed GUI makes advanced LLM interpretability tools accessible to everyone. With intuitive features for dataset selection, model setup, visualization, and experimentation, it simplifies the process for non-technical users. Future work will explore additional features such as real-time collaborative editing and integration with cloud-based model training services.

6 Acknowledgment of Design Tools

The following drawings were created using Figma. You can view the designs at this link: CBM-UI Design in Figma.

7 References

Sun, C.-E., Oikarinen, T., and Weng, T.-W. Crafting large language models for enhanced interpretability. In ICLR, 2024.

8 Appendix

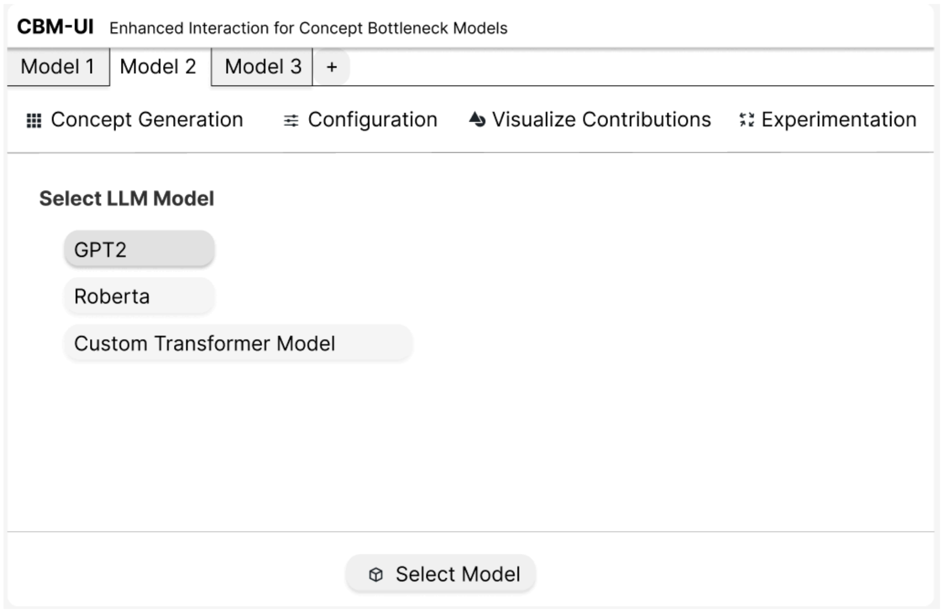

Figure 5: Model Selection Interface

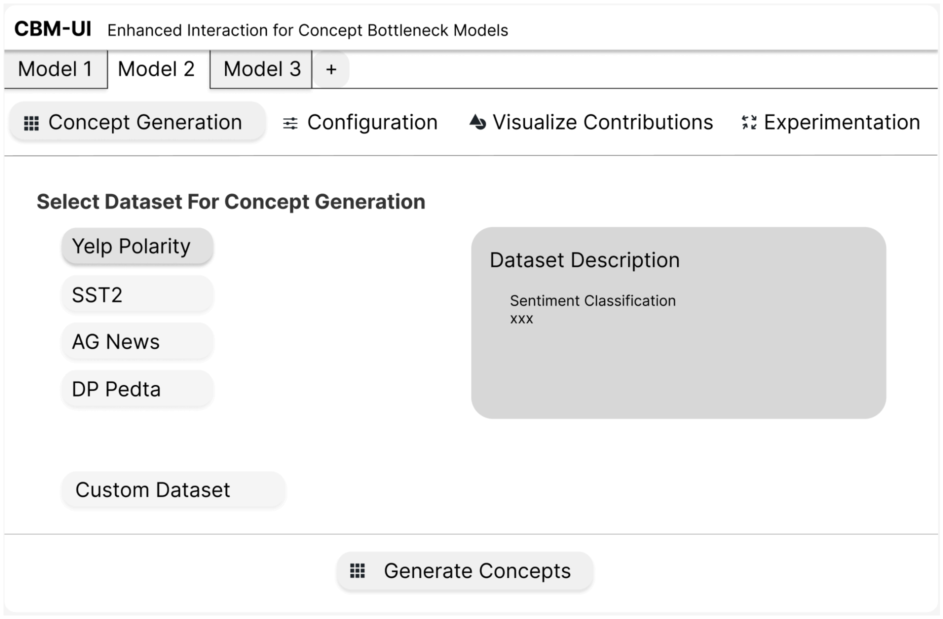

Figure 6: Concept Generation Section

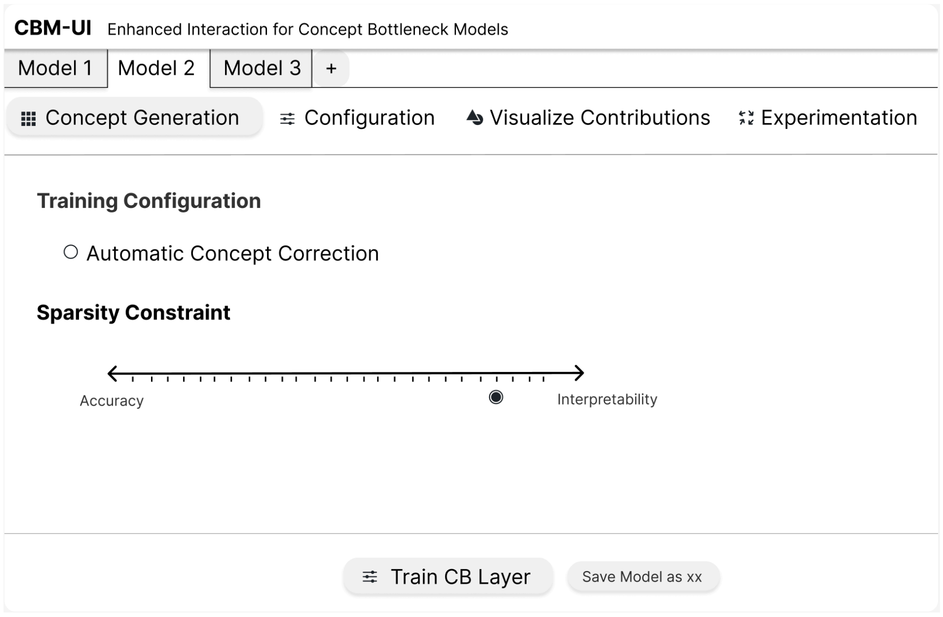

Figure 7: Model Configuration

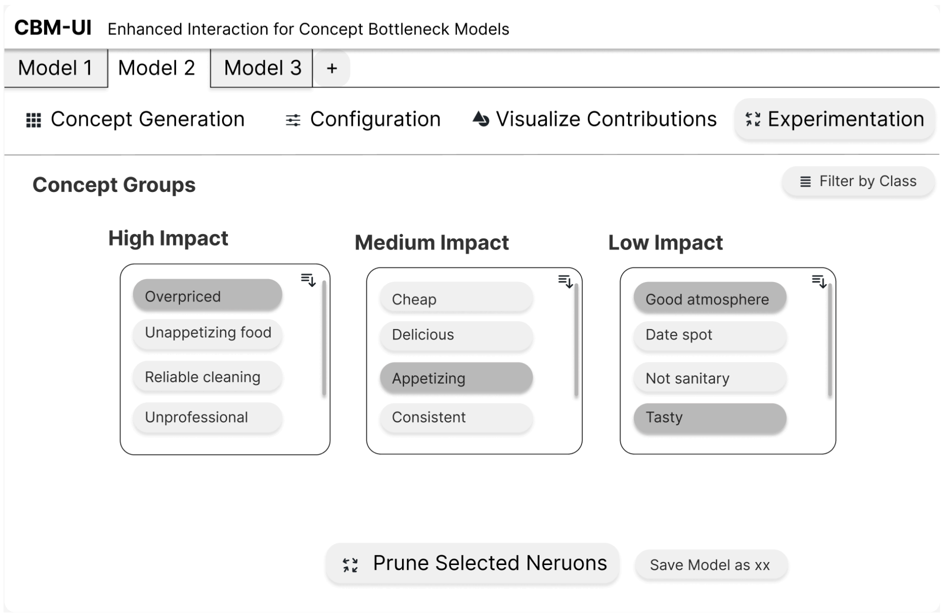

Figure 8: Experimentation Page. Users can select concepts to prune.

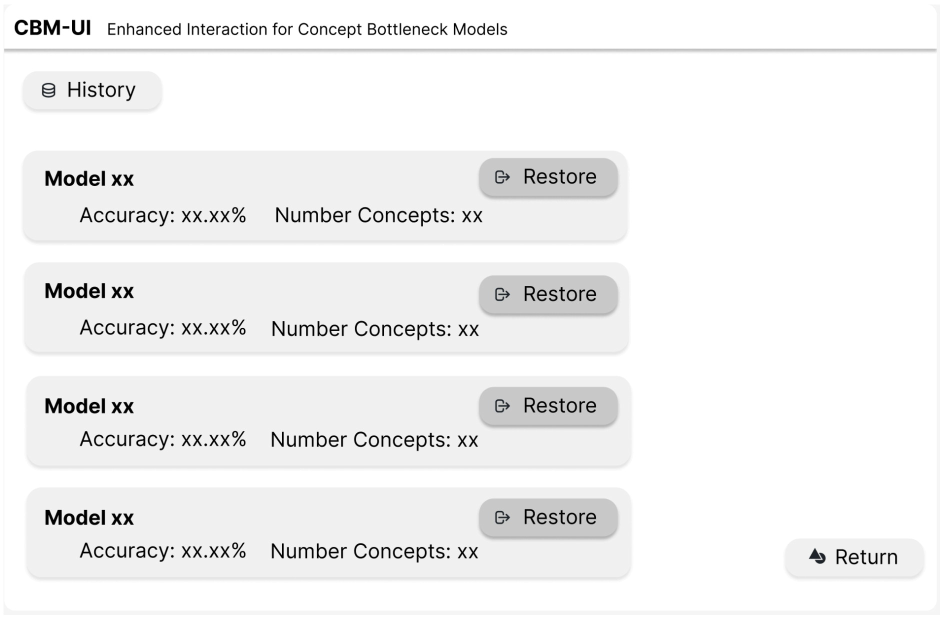
CBM-UI Enhanced Interaction for Concept Bottleneck Models
History
Model xx
Restore
Accuracy: xx.xx% Number Concepts: xx
Model xx
Restore
Accuracy: xx.xx% Number Concepts: xx
Model xx
Restore
Accuracy: xx.xx% Number Concepts: xx
Model xx
Restore
Accuracy: xx.xx% Number Concepts: xx
Return

Figure 9: History Page. Users can choose to restore previous models.