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

**Gabriel Cha** gcha@ucsd.edu
**Jiarui Zha** jzha@ucsd.edu
**Steven Luong** sluong@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). The GUI supports tasks such as dataset selection, model configuration, concept visualization, and experimenting with concept pruning. This work builds upon the research by Weng et al. (2024).

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

1 Background and Related Work . . . 2
2 Tech Stack . . . 3
3 Model Selection Tab . . . 3
4 Features . . . 3
5 Conclusion . . . 5
6 Acknowledgment of Design Tools . . . 5
7 References . . . 5
8 Appendix . . . 5
References . . . 5

# 1 Background and Related Work

LLMs are powerful but difficult to interpret. CBMs improve interpretability by linking neurons to human-understandable concepts. Weng et al. (2024) developed an automated pipeline to integrate CBLs into LLMs, but it requires users to run technical scripts. Our project simplifies this process by building a user-friendly GUI and container, making setup seamless and accessible, guiding users through dataset selection, model configuration, concept visualization, experimentation, and history management.

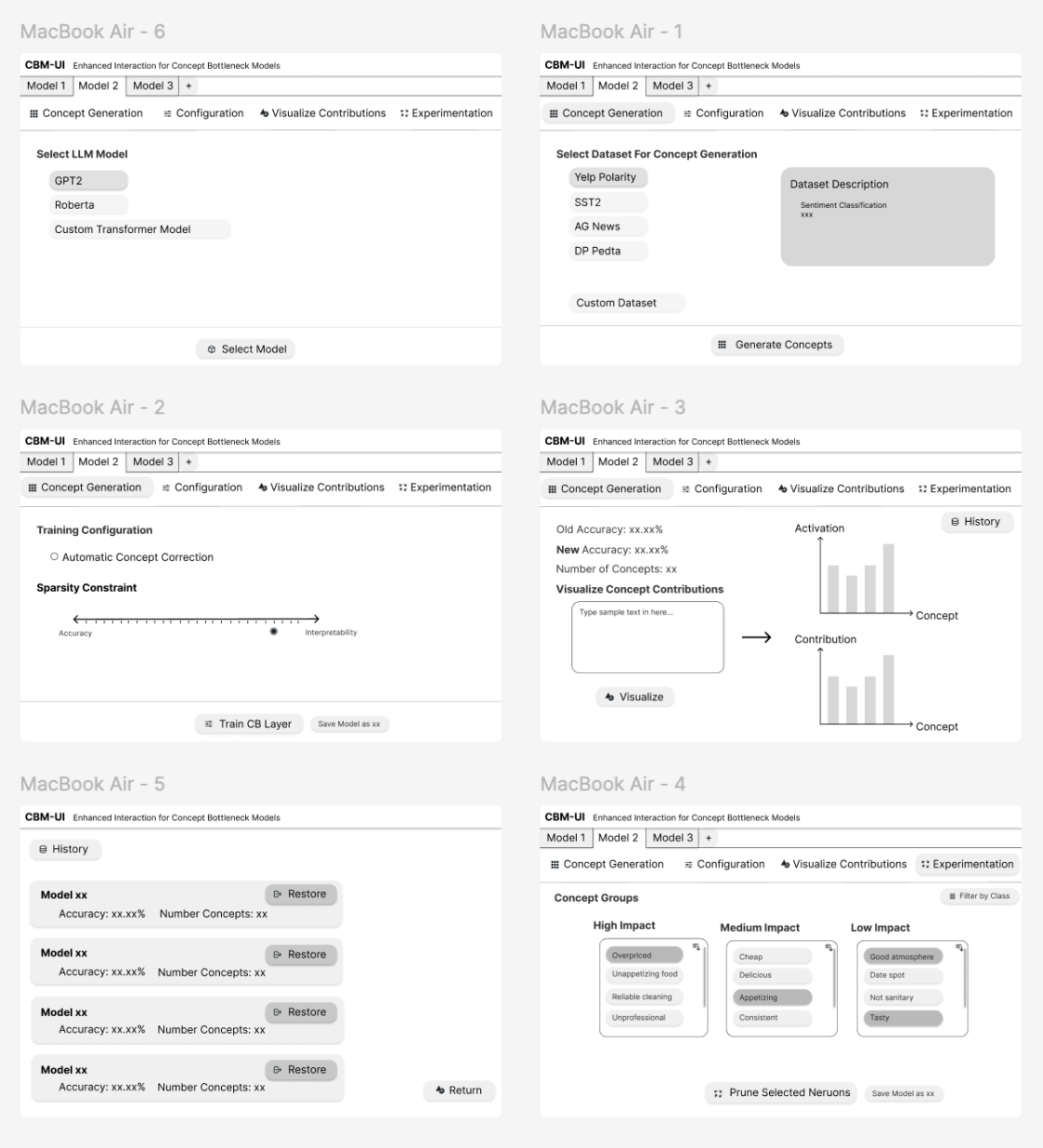


Figure 1: Overview of the CBM-GUI Workflow: This figure illustrates the main interfaces of the Concept Bottleneck Model (CBM) Graphical User Interface.

# 2 Tech Stack

The proposed GUI will use React for the frontend and Python for the backend. React will handle the dynamic UI, while Python will run ML tasks using PyTorch. When a user clicks a button, React will send a request to the Python backend, triggering a script. The backend will process the request, and React will update the UI with the results.

# 3 Model Selection Tab

When users enter the page, they are presented with options to select or add models for their tasks. They can choose from preloaded models like GPT or RoBERTa, or upload their own pre-trained model in `.pt` format. This provides flexibility, allowing users to work with a variety of models or integrate their custom models seamlessly. As shown in Figure 3 in the appendix, the Model Selection Interface provides a clear and user-friendly way to manage model options.

# 4 Features

## 4.1 1. Dataset Selection

Users can pick from preloaded datasets (e.g., Yelp Polarity, SST2) or upload their own. A description panel helps them understand each dataset's purpose. **User Flow**: After selecting a dataset, users proceed by clicking "Generate Concepts," which triggers the backend process of extracting relevant concepts linked to the dataset. See Figure 4 in the appendix.

## 4.2 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 5 in the appendix.

## 4.3 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.

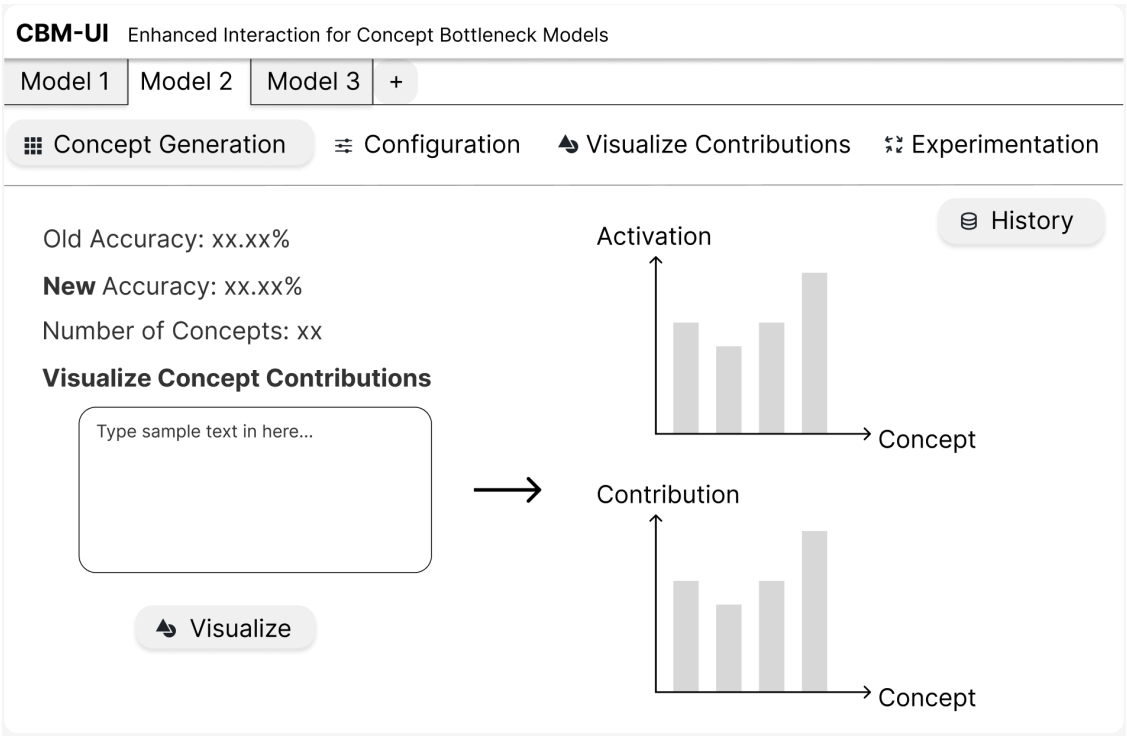


Figure 2: Users can compare how different configurations affect the model's decision-making process by interacting with the visualization tools.

## 4.4 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 6 in the appendix.

## 4.5 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 7 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

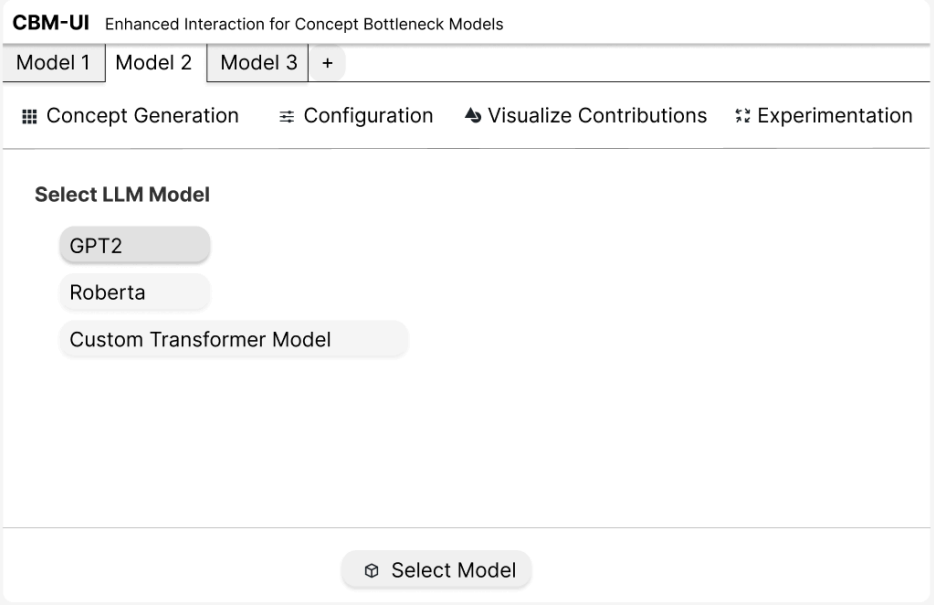


Figure 3: Model Selection Interface

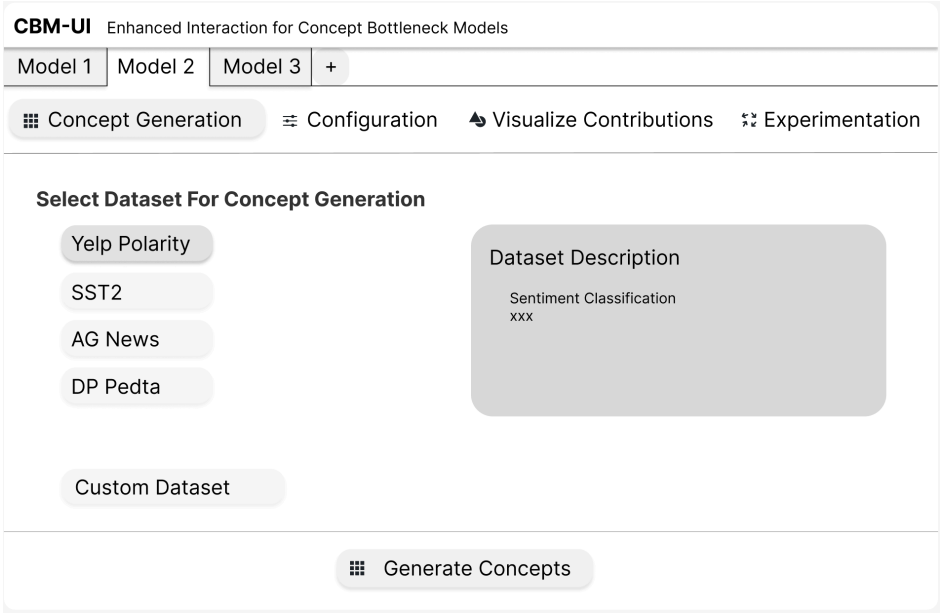


Figure 4: Concept Generation Section

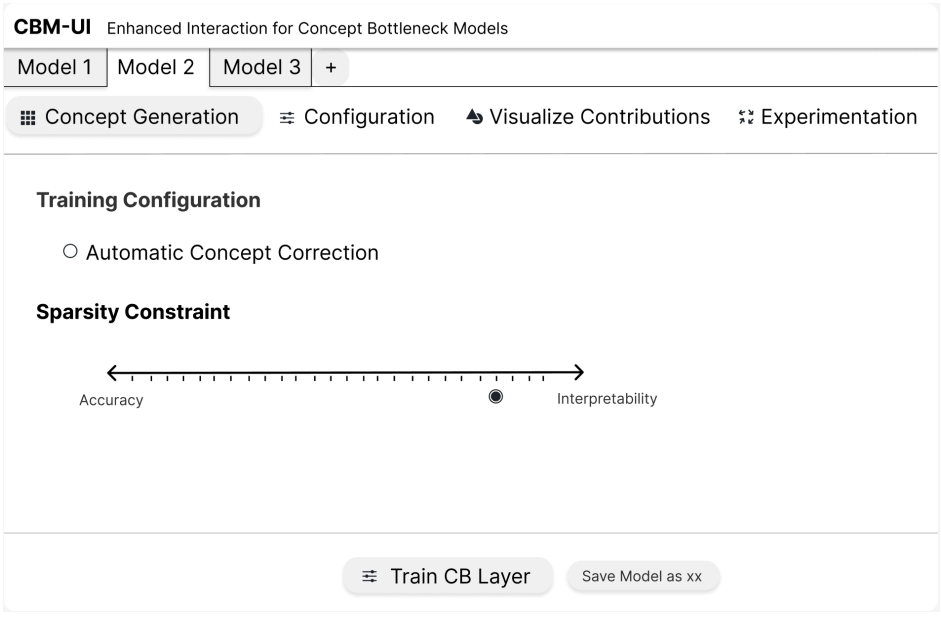


Figure 5: Model Configuration

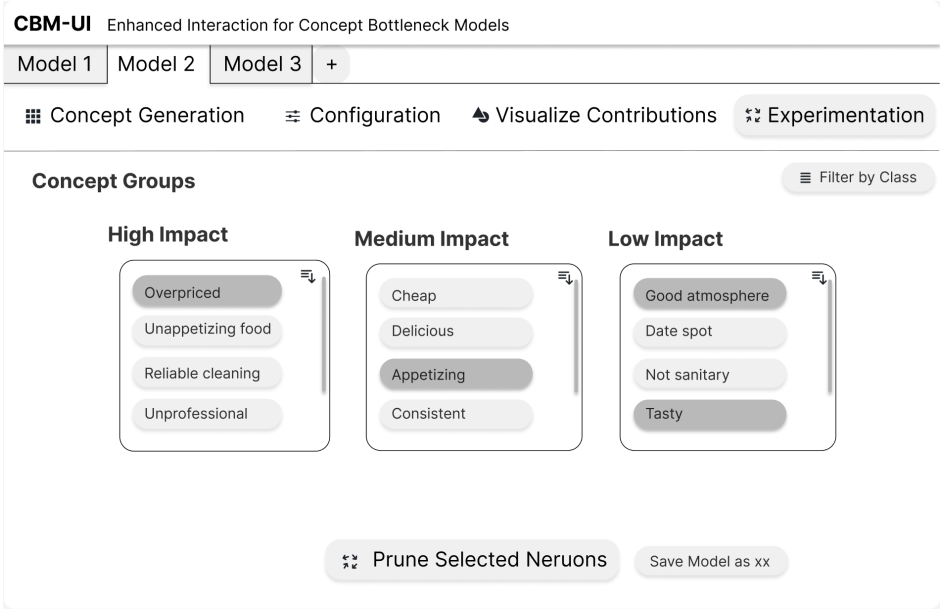


Figure 6: Experimentation Page. Users can select concepts to prune.

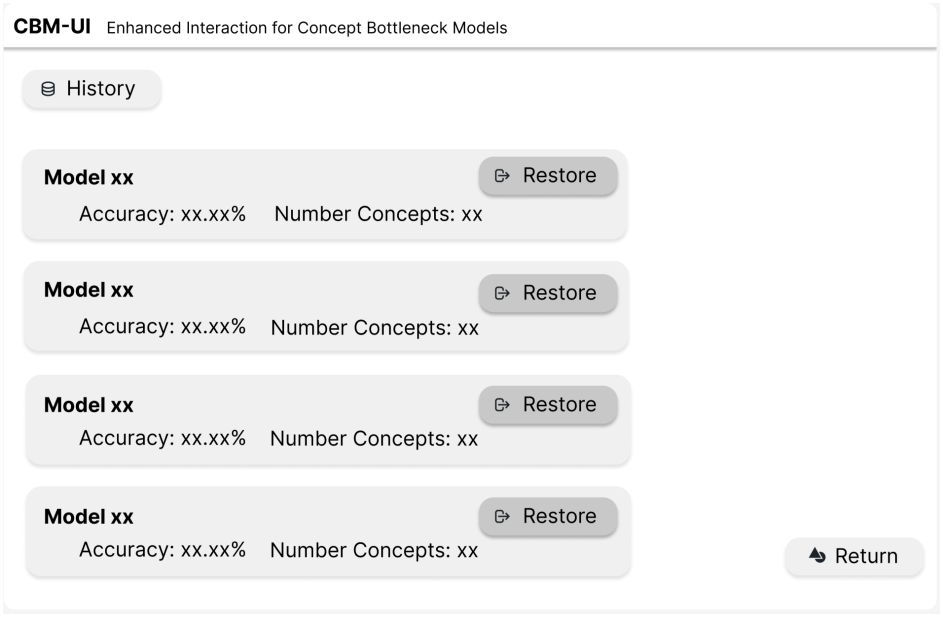


Figure 7: History Page. Users can choose to restore previous models.