Toward Trustworthy Learning-Enabled Systems with Concept-Based Explanations

Sagar Patel
University of California, Irvine

Dongsu Han
KAIST

Nina Narodystka
VMware Research by Broadcom

Sangeetha Abdu Jyothi
University of California, Irvine
VMware Research by Broadcom

Abstract

Despite the superior performance of deep learning-based controllers in network applications, their practical adoption is limited due to the difficulty in understanding and trusting them. Existing explainability solutions largely focus on interpreting these controllers by providing insights into the top features used by the model. Although these insights can help reveal an important aspect of the controller, they require operators to deal with low-level features, requiring extensive manual analysis and interpretation.

In this work, we present a novel explainability approach that provides insights to operators using high-level, human-understandable *concepts* (e.g., 'fluctuating network throughput'). Our approach offers an intuitive platform for operators to identify unintended behaviors, develop strategies to address them, and define data collection strategies to implement them. Our concept-based explainability framework lays the foundation for an intelligent AI system where operators can design the controller they intend using familiar terminology and domain knowledge. We provide an initial implementation of our ideas in adaptive video streaming and demonstrate its potential.

CCS Concepts

• **Computing methodologies** → **Machine learning**; • **Networks** → *Application layer protocols.*

Keywords

Machine Learning for Systems, Explainability

ACM Reference Format:
Sagar Patel, Dongsu Han, Nina Narodystka, and Sangeetha Abdu Jyothi. 2024. Toward Trustworthy Learning-Enabled Systems with Concept-Based Explanations. In *The 23rd ACM Workshop on Hot Topics in Networks (HOTNETS '24), November 18–19, 2024, Irvine, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3696348.3696894

1 Introduction

Today, learning-based controllers outperform manually designed ones in a range of computer systems and networking applications, delivering significant performance gains in applications including adaptive bitrate streaming [14, 25], congestion control [5, 11, 26], resource management [29], network traffic optimization [9], edge caching [21], and network security [20]. However, operators are often reluctant to deploy these solutions because they are difficult to interpret, debug, and trust [15].

Current explainability solutions for learning-based controllers focus primarily on interpreting the decision by identifying the most influential input features. For example, Metis [15] employs a model distillation technique to convert the deep learning model into a decision tree and presents feature-level decision paths as explanations. Building on this approach, Trustee [10] improves fidelity through decision tree pruning and generates a trust report. However, these feature-based explainability solutions have intrinsic limitations.

Feature-based explanations typically involve large decision trees with hundreds of nodes and complex decision paths. For example, the video streaming explainer generated using Trustee in Fig. 1b has a decision path [buffer$_{t-1} \leq 0.91$; chunk size$_{t-1} \leq 0.05$; past quality$_{t-1} \leq 0.66$; ...]. Such rule-based interpretations of the controller, while more meaningful than a black-box neural network, are practically difficult to interpret as they involve over a dozen decision points across a disparate set of features. More importantly, existing feature-based explanation techniques [13, 19] fail to capture high-level concepts involving complex combinations of inputs. For example, to identify volatile network conditions, one has to carefully analyze the throughput decision points across different time instances.

In this paper, we propose *concept-based explanations* as a suitable technique to tackle multifaceted explainability challenges in learning-enabled systems. *Concepts* are high-level, human-understandable attributes that encapsulate complex controller and environment characteristics, typically integrating multiple input features. This abstraction allows us to succinctly capture intricate patterns, trends, and behaviors in systems. For instance, in video streaming, concepts might include 'Anticipation of congestion' or 'Rapidly depleting buffer' rather than raw throughput and delay thresholds presented by feature-based explainers. By using concepts as the fundamental units of explanation, we are taking a step towards describing the controller's decision-making process in a manner that humans naturally think about the system. This approach will significantly enhance the operator's ability to understand, validate, and trust the model's reasoning process across the different stages of the system lifecycle.

Self-interpretable Concept-Bottleneck Models (CBMs) used in computer vision [12, 27, 28] cannot be directly adapted to systems environments due to several reasons: (i) Systems controllers take as inputs heterogeneous unlabeled data, while vision models handle structured images, with a wide availability of labeled image datasets. (ii) Concepts in learning-enabled systems are more complex, usually requiring multiple levels of abstraction, even for domain experts to capture system behaviors, unlike visual features. (iii) Systems controllers typically handle complex temporal dynamics and long-term dependencies. Consequently, a new approach to concept-level understanding is required in learning-enabled systems.

We design an LLM-driven concept-based explainability framework to address the unique challenges in learning-enabled systems. At a high level, our framework first uses a Large Language Model (LLM) [6] to derive a set of base concepts, for example, from survey papers in the target domain. To enrich and customize the concepts, we envision augmenting them with operator-defined ones. Next, for each input state, we leverage the LLM to generate a concept-based description of the state using a pre-crafted prompt. Then, we embed the base concepts and the state's description using a text embedding model [3, 24] and obtain empirical data mapping the controller's inputs to the well-defined concepts in the literature. Finally, we use this data and train the two functions of our explainer: the concept-mapping function, which identifies the concepts present in the inputs, and the output-mapping function, which linearly combines them to generate the controller's output.

Our concept-based approach unlocks new capabilities by addressing distinct explainability needs at each stage of the system lifecycle—from design and testing to deployment and retraining. During the *design phase*, a concept-based explainer can ensure that operator intent is effectively captured in the trained model by aligning high-level concepts with operator goals. During *testing*, the explainer facilitates comprehensive coverage and robust performance validation across scenarios. During *deployment* in production environments, the explainer will play a key role in deciphering complex behaviors beyond primary input features. Finally, during *retraining*, the explainer can help pinpoint underspecified data and guide refinement by identifying conceptual gaps in the model's understanding. Our framework is generalizable across controllers trained using supervised, semi-supervised, and reinforcement learning.

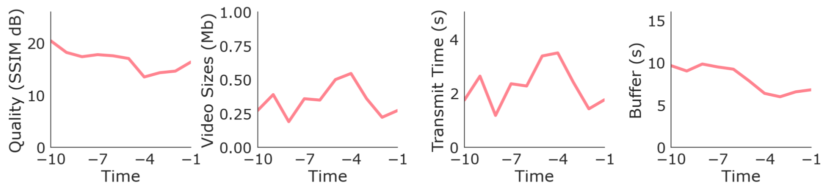
To demonstrate the benefits of this approach, we prototype our idea using adaptive bitrate streaming as a running example and demonstrate its capabilities in testing and retraining. We show that operators can intuitively (i) identify causes for unintended behavior using conceptual reasoning and find mitigation strategies to correct such behaviors, and (ii) define data-collection strategies to implement the mitigation strategies during retraining. Finally, we discuss how the idea can be extended to realize the full potential of explainability during all phases of the system lifecycle.

2 Motivation

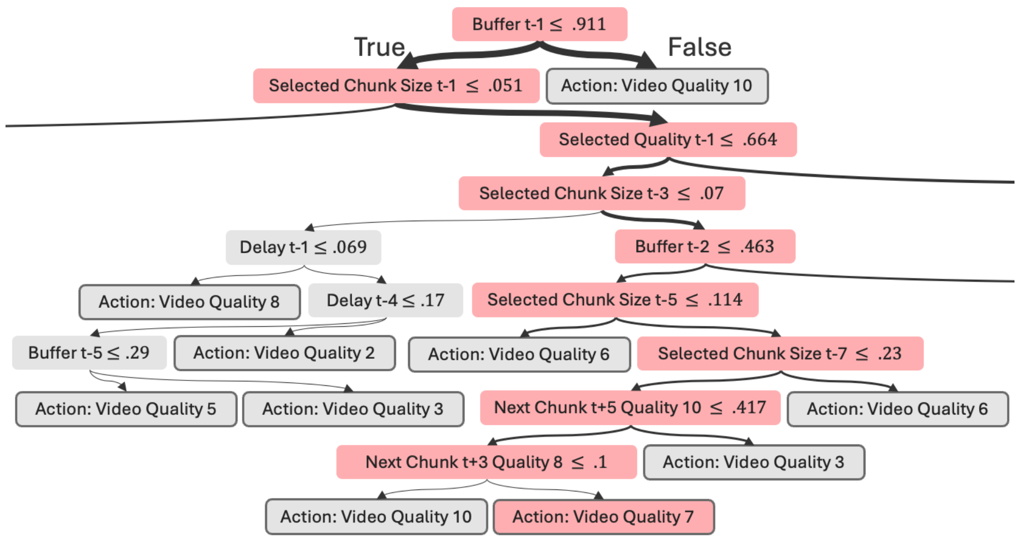
We highlight the challenges with prior explainability techniques using an adaptive bitrate (ABR) streaming controller as a representative example. In online video streaming, the video is divided into short chunks and encoded at multiple bitrates. The ABR controller adaptively selects between these bitrates in response to the network conditions while maximizing the viewer's Quality of Experience (QoE).

We use Gelato [16], the publicly available state-of-the-art controller on the streaming platform Puffer [25], delivering online live television to over 280,000 users [4, 25]. Gelato is a Deep Reinforcement Learning controller and takes as input the history of the stream and information about the upcoming video chunks. The history includes six features: the selected video quality, the size of the selected chunk, the time taken to transmit that chunk, the video buffer occupancy of the client, obtained QoE, and the video stalls. The information about upcoming chunks includes the video sizes and qualities at every encoded bitrate for the next five timestamps. Gelato then outputs the bitrate to send next.

Motivating question. We highlight the challenges of feature-based explainer using a hypothetical scenario. The operator seeks to understand the behavior of the controller when it



(a) The motivating state



(b) The decision path of the state in the pruned report

Figure 1: Trustee [10]'s explanation for the motivating state. Both the full and pruned decision trees are large in the Trustee report (195 and 65 nodes, respectively). Even filtering just one decision path for the motivation state leads to a complex explanation.

picks one of the medium quality bitrates in response to a decreasing but half-full buffer, visualized in Fig. 1a.

Feature-level explainers' answer. Trustee [10] generates a global explanation of the controller as a decision tree. The complete tree has 195 decision nodes and a depth of 13. Even the pruned version of the tree used in the Trustee report has 61 nodes and a depth of 10. We generate an explanation for the motivating scenario by traversing the decision path on the pruned tree (Fig. 1b). We find that the explanation is difficult to understand, involving multiple decision nodes on the features—buffer, the size of the selected chunk, and upcoming video qualities—all split across multiple timestamps. This explanation suggests that the controller uses some trend in the history and future chunks to determine the next bitrate but does not explicitly reveal how it does so. Feature-level explanations force human operators to closely understand the intricacies of the feature set originally designed for a machine learning algorithm, which often proves difficult to use without extensive expertise and manual effort. This inherent limitation is shared by all feature-based explainers [13, 19].

3 Towards High-level Explainability

In this section, we provide a brief overview of concept-based explanations and present our proof-of-concept design tailored for learning-enabled systems. Our concept-based framework can be easily integrated with deep learning system controllers trained using supervised, semi-supervised, and reinforcement learning paradigms.

3.1 Concept-Based Explainability

Concept Bottleneck Models [12, 27, 28], recently proposed in computer vision, use human-understandable concepts as intermediaries between input data and model output. They transform arbitrary intermediate output to high-level concepts (e.g., 'beak' and 'wings' for a bird image) and then map these to final outputs. This allows their output to be interpreted as a weighted sum of high-level concepts. The training process uses a small set of model input-output samples and concept-level tagging. While vision tasks involve simple concepts easily tagged by text-to-image generative

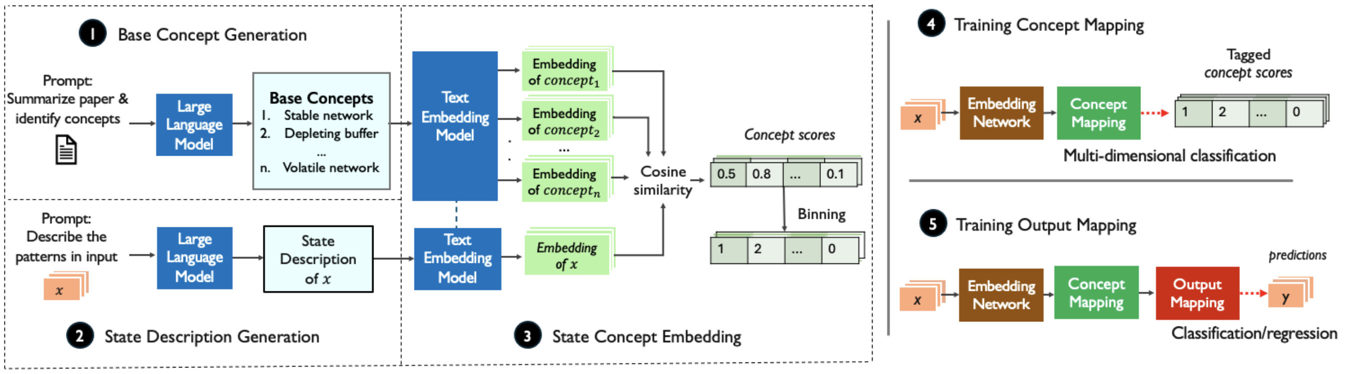


Figure 2: Training pipeline of the concept-based explainer. We generate the base concepts and the description of input states using LLM queries. Next, we embed the base concepts and the state description using a text embedding model, and generate concept scores based on their similarity. Then, we sequentially train the concept mapping and output mapping functions using the concept scores and the target output. The red dashed arrows show the learning.

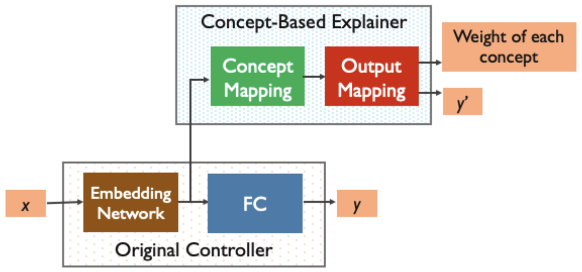


Figure 3: Inference on the concept-based explainer.

models [17, 18] or crowdsourcing [1], networking applications present more complex concepts (§ 2). These require deep system understanding and reasoning over multiple features across time, making simple tagging methods ineffective.

3.2 Concept-Based Explanations for Systems

We propose the first practical framework for concept-based explanations in learning-enabled systems. An explanation in our framework is a linear combination of a pre-defined set of base concepts. We detail the training and inference on this framework.

3.2.1 ***Training the Concept-based Explainer.*** We show the training workflow in Fig. 2. Our training pipeline has five key stages. ❶ *Base concept generation*: We first generate the set of base concepts using an LLM query to summarize a survey paper in the target domain. ❷ *State description generation*: We use an LLM query with the input state of the controller and a pre-crafted prompt, which includes the base concepts, to generate a concept-based state description. ❸ *State Concept Embedding*: We embed the base concepts and the state description using a text embedding model and measure their similarity to identify the dominant concepts. ❹ *Training Concept mapping*: We learn a mapping from the controller's embedding to concept classes. ❺ *Training Output mapping*: Finally, we learn a mapping from the concept space to the controller output space. With these two functions together, we can backtrack the controller's decision-making process and observe the dominant concepts driving the output.

Base Concept Generation. The base concepts serve as the units of explanation of the concept-based explainer. The controller's output will be explained as a linear combination of these concepts. Defining a comprehensive set of concepts that capture all the reasons behind the controller's decision-making at the appropriate level of abstraction is challenging. We find that simply querying the state-of-the-art LLMs [6, 8, 23] for base concepts does not produce a usable set. Hence, we opt for an alternative approach: we first ask the LLM to summarize the findings of a relevant survey paper. We then use this context and prompt the LLM to identify relevant concepts, along with an explanation for its selection. This approach leads to a set of succinct base concepts with high coverage of relevant concepts. The base concepts generated for the motivating ABR scenario (§ 2) using survey paper [7] are shown in Table 1. These base concepts may be augmented with operator-defined ones.

State Description. Input states in networked systems have complex patterns like 'network throughput volatility,' which can be difficult to label, even for experts. Hence, rather than directly attempting to tag the concepts in the input states, we first get a description of the state and then identify the concepts with that description. We use a pre-crafted LLM prompt that includes the state variables, their values, and the base concepts generated in the previous stage. The current prototype relies on a single query per state; multiple LLM queries may be combined to improve accuracy in the future.

State Concept Embedding. Next, we generate the concept embedding in two steps. First, we embed the base concepts and the state description using a text embedding model [3, 24]. Second, we measure the similarity between the base concepts and the state in the text embedding space using the cosine similarity metric. We bin the similarity concepts into one of k classes at this stage. The process can be represented mathematically as follows:

$$Sc_c = \psi_k \left(\frac{\epsilon(x) \cdot \epsilon(c)}{\|\epsilon(x)\| \, \|\epsilon(c)\|} \right) = \psi_k(\epsilon(x) \cdot \epsilon(c)), \forall c \in C. \quad (1)$$

Here, Sc_c is the score of each concept c in the set C, ϵ is the text embedding model, x is the controller input, and ψ is the quantization function.

Training Concept Mapping. During training, we use examples from a small subset of x to obtain controller embeddings $h(x)$ and predict the concept classes Sc_c for all $c \in C$. The multi-label classification loss function used is:

$$l(x, Sc) = \sum_{c}^{C} \left[-\log \left(\frac{e^{(\delta_\theta(h(x)), Sc_c)}}{\sum_{i=1}^{k} e^{(\delta_\theta(h(x)), i)}} \right) \right] \quad (2)$$

We minimize this loss using mini-batch stochastic gradient descent to optimize the parameters θ of the concept-mapping function δ_θ. The output of δ_θ is a $C \times k$ matrix showing the probability of each bin for each concept, where $\delta_\theta(h(x)), Sc_c)$ is the element at index (c, Sc_c). Importantly, the gradient does not propagate back to x, keeping the controller neural network fixed.

Training Output Mapping. In the final stage, we learn the linear output-mapping function to ensure that the explainer closely tracks the controller by transforming the concept space back to the controller's output space. Using examples from a small subset of x, we obtain the concepts $\delta_\theta(h(x))$ and predict the controller output y. We find the weight matrix W and bias $\vec{b}$ of function $y = W^T(\delta_\theta(h(x)) + b$ using mini-batch stochastic gradient descent, with mean squared error for continuous outputs or cross-entropy for discrete ones. However, there is a fidelity-complexity trade-off: weights and biases that match the controller's output with high fidelity may rely on a large number of concepts. To balance this, we apply ElasticNet regularization [30] to W and $\vec{b}$:

$$l_{elastic} = (1 - \alpha) \|W\|_2^2 + \alpha \|W\|_1 + \alpha \|b\|_1 \quad (3)$$

Note that we perform the learning process of concept mapping and output mapping sequentially and independently. The concept-mapping function does not influence the controller output, and the output-mapping function does not affect the concept space. This separation prevents any gradient leakage between the functions and ensures high accuracy.

1. Volatile Network Throughput	9. Stable Buffer
2. Rapidly Depleting Buffer	10. Nearly Full Buffer
3. Low Content Complexity	11. Startup of video
4. Recent Network Improvement	12. High Content Complexity
5. Extreme Network Degradation	13. Network Volatility needing Switching
6. Moderate Network Throughput	14. Avoiding Large Quality Fluctuations
7. Anticipation of Network Congestion	15. Switch to higher quality after startup
8. Content requiring High Quality	16. High Network Throughput

Table 1: Base Concepts in Adaptive Bitrate Streaming.

3.2.2 *Inference on the Concept-based Explainer.*

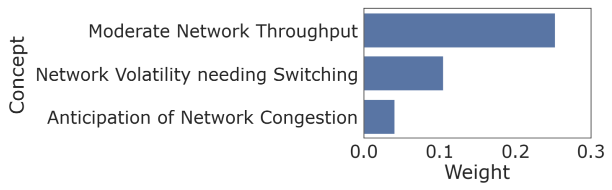
The inference relies on the embedding network of the original controller and the trained concept mapping and output mapping functions (Fig. 3). Note that LLMs are not used in inference since the concept mapping function is already trained to map the controller's embedding to the concept space. Hence, inference is cheap and efficient.

The inference API of our concept-based explainer currently supports three types of queries: (i) *Factual single state*: Generates an explanation based on the top concepts in the given state. (ii) *Counterfactual states*: Generates an explanation for how state$_A$ differs from state$_B$ by comparing the top concepts in both states. (iii) *Counterfactual actions*: Generates an explanation for why the controller chose action Y_1 over action Y_2 by comparing the top concepts in the two actions.

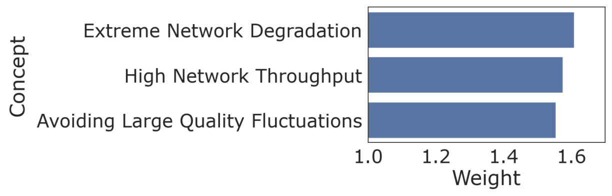
4 Case Study: Adaptive Bitrate Streaming

We demonstrate the capabilities of our explainer with ABR as a representative application. We return to the motivating case in §2 to show its ability to handle complex scenarios. We use the Large Language Model GPT-4o [2] and the text embedding model OpenAI Large [3] in our training pipeline. We again focus on the DRL controller Gelato [16], the state-of-the-art controller on the live streaming platform Puffer [25]. We use the same subset of controller input-output samples the operator used to generate the Trustee [10] explanations, a set of 4,000 states and actions obtained by rolling the controller out using Puffer traces.

Quality of Explanations. Following the convention of prior explainability approaches [10, 15], we measure the quality of our explanations using fidelity, which represents how well the explanation model replicates the output of the original neural network. We rollout the controller on a held-out set of Puffer traces and create a set of test input-output samples (x, y) on the original controller. Then, we query the learned



(a) Concept-level Explanation for medium quality



(b) The key concepts towards a higher quality

Figure 4: The high-level concept-based explanation answering the Motivation Question. The explanations are able to both identify the reason the controller chose the mediun quality bitrate and show the concepts needed to push to a different bitrate, the high quality one.

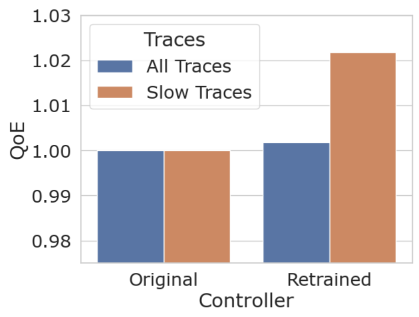


Figure 5: Performance of the controller after being retrained with the data-collection strategy defined by the operator, focusing on the concept 'Extreme Network Degradation'.

concept-based explainer and evaluate the accuracy of its predicted y' to the controller's output y. This accuracy indicates how well the explainer is tracking the original controller. We observe that concept-based explanations are highly accurate, achieving an accuracy of 98% surpassing the previous state-of-the-art Trustee technique [10], which had 94% accuracy. This underscores the power of concept-based explainers, offering a path beyond the strict fidelity-explainability tradeoffs of feature-level explainers, achieving high fidelity without the complexity of deep trees.

While fidelity is important, it does not measure the utility of explanations in practical applications. Therefore, we now turn to demonstrate how concept-level explanations can intuitively enable practical explainability applications.

Identifying unintended behaviors during testing. Returning to the motivating scenario, we generate a factual state explanation for the medium-quality bitrate when the buffer was decreasing but still half-full (Fig.1a). Our concept-based explanation, visualized in Fig.4a, reveals that the controller's action was primarily due to 'Moderate Network Throughput', 'Network Volatility needed Switching' and 'Anticipation of Network Congestion'. While this reasoning can largely be appropriate, the operator believes the controller may still be too conservative and seeks strategies to improve the Quality of Experience (QoE) in such scenarios.

Finding strategies to change the controller. To find a path to achieve the desired change, the operator queries the concept-level explainer again with the counterfactual action query for a higher bitrate in the scenario. The results, visualized in Fig. 4b, suggest that 'Extreme Network Degradation' is a key factor that needs to be prioritized. This suggests that improving the controller's response to worsening conditions can enhance the QoE during retraining.

Defining data-collection strategies to implement the path. Having identified 'Extreme Network Degradation' as the focus concept, the operator seeks to implement strategies for the controller to collect more data in that scenario. For adaptive bitrate controllers such as Gelato, this means collecting more experience interacting with the traces that exhibit this scenario. To do so, we tag each trace with the top three concepts of all its component states. The operator then rolls out the controller in the training traces with the relevant concepts. In this example, they retrain the controller, simply focusing solely on the traces with 'Extreme Network Degradation' as a dominant concept.

In Figure 5, we benchmark the controller's performance before and after the operator's intervention. The retrained controller, having collected more data on 'Extreme Network Degradation', outperforms the original, achieving significantly higher QoE in slow traces while maintaining similar performance across all traces. This demonstrates how straightforward the debugging and design process can be, requiring the operator to assess only the high-level reasoning which is already part of their analysis.

Scenario-level Evaluation. Networking applications involve performance trade-offs where improvement in one scenario leads to a decline in another. Our concept-based explainer can augment commonly-used throughput/latency profiling of workloads [14, 22, 25] with fine-grained concept-level profiling. Fig.6 demonstrates this approach, revealing that retraining yields uneven performance gains: significant improvements in 'Extreme Network Degradation' and 'Stable Buffer' conditions but slight declines in 'Rapidly Depleting Buffer'. Combined with LLMs, these explanations can streamline the evaluation process and help operators answer

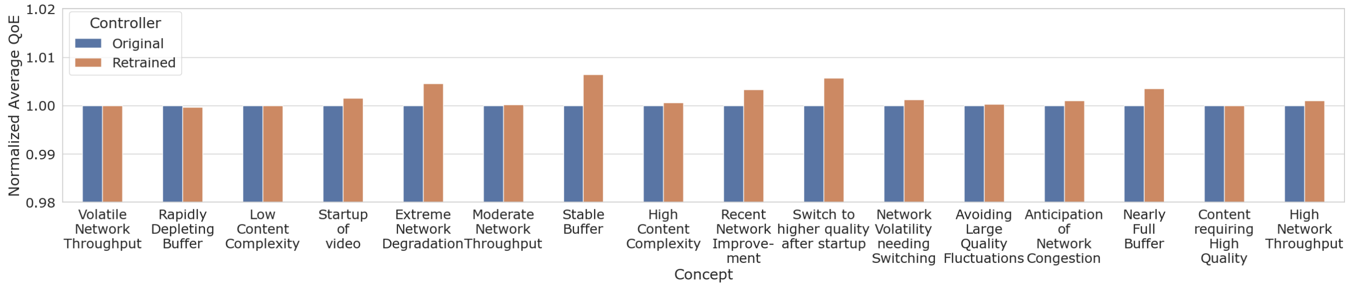


Figure 6: Performance of the original and retrained controllers measured across high-level concepts.

questions such as, "With this change, does the controller improve at handling low-bandwidth users in country A?"

5 Future Directions

Our case study on ABR serves as an initial demonstration of the potential applications of high-level explainability in learning-enabled systems. We now discuss our high-level vision of what concept-based explainers could enable.

Interactive Intent-Driven Controller Design. We envision that concept-based explanations would enable a breakthrough in data-driven controller design, allowing operators to identify problems and interact in natural language to guide the controller behavior to match the operator's intent. This could enable an iterative process where the operator interacts with an LLM, which would then formulate an explainer query, run it, and provide the operator with the answer needed to decide the next steps.

Mapping input state to concepts, we believe, is a step towards Artificial General Intelligence (AGI) enabled controller design as it connects the inputs and output of a controller to a complex and nuanced encoding of concepts, semantics, and relationships. We believe that logic and ideas from existing literature often embodied in LLMs can be combined with model-specific, data-driven learning to create more intelligent controllers. Realizing the idea fully requires further research.

Steering Dataset Selection during Design. Concept-based evaluation of datasets can offer insights into training data's conceptual coverage and diversity. This analysis can guide the selection of additional datasets by identifying conceptual gaps in the current training data, thereby improving the controller's exposure to a comprehensive range of relevant concepts. Such informed dataset selection could lead to more robust and generalizable controllers.

Guiding Architectural Modifications during Design. Neural architecture issues can cause controller misbehavior [15]. Concept-based explanations, combined with neural network analysis techniques such as Saliency Maps, can help map high-level concepts to specific parts of the neural network and, thus, facilitate model redesign with critical reasoning and observability.

Defining Concept Coverage Metrics for Testing. Drawing parallels with code coverage in software testing, we can develop concept coverage metrics for testing learning-enabled systems. These metrics would quantify the extent to which a test suite exercises different concepts relevant to the system's operation, revealing conceptual blind spots. By ensuring that a diverse range of concepts is adequately tested, these metrics can help improve both the robustness and reliability of the learning-based system.

Online Safety Assurance in Deployment. Concept-based explainers can be used to monitor dominant concepts in the input data stream of the controller and trigger switching to a heuristic when concepts that the controller cannot handle are observed in the input data. Thus, they can help ensure the system remains safe under uncertain or novel conditions.

6 Conclusion

In this paper, we present a radically new explainability framework designed for human operators based on high-level concepts. Through case studies, we demonstrate that our framework can enable operators to intuitively perform key explainability use cases, such as identifying potential unintended behaviors using conceptual reasoning and designing strategies to mitigate them. We hope this ease can highlight the potential of high-level concept-based explanations and offer insights to one day power a system where operators design and debug controllers through their domain expertise using natural language.

Acknowledgments

We thank the anonymous reviewers for their insightful feedback and Jinwoo Shin for generously sharing his expertise on concept bottleneck models. This work is supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) of the Korea government (MSIT) (No. RS-2024-00418784).

References

[1] [n. d.]. Amazon Mechanical Turk. https://www.mturk.com/. (Accessed on 06/16/2024).

[2] [n. d.]. Hello GPT-4o | OpenAI. https://openai.com/index/hello-gpt-4o/. (Accessed on 06/26/2024).

[3] [n. d.]. New embedding models and API updates | OpenAI. https://openai.com/index/new-embedding-models-and-api-updates/. (Accessed on 06/17/2024).

[4] [n. d.]. Puffer. https://puffer.stanford.edu/results/. (Accessed on 04/20/2022).

[5] Soheil Abbasloo, Chen-Yu Yen, and H Jonathan Chao. 2020. Classic meets modern: A pragmatic learning-based congestion control for the Internet. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication.* 632–647.

[6] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[7] Abdelhak Bentaleb, Bayan Taani, Ali C Begen, Christian Timmerer, and Roger Zimmermann. 2018. A survey on bitrate adaptation schemes for streaming media over HTTP. *IEEE Communications Surveys & Tutorials* 21, 1 (2018), 562–585.

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[9] Li Chen, Justinas Lingys, Kai Chen, and Feng Liu. 2018. Auto: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization. In *Proceedings of the 2018 conference of the ACM special interest group on data communication.* 191–205.

[10] Arthur S Jacobs, Roman Beltiukov, Walter Willinger, Ronaldo A Ferreira, Arpit Gupta, and Lisandro Z Granville. 2022. Ai/ml for network security: The emperor has no clothes. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security.* 1537–1551.

[11] Nathan Jay, Noga Rotman, Brighten Godfrey, Michael Schapira, and Aviv Tamar. 2019. A deep reinforcement learning perspective on internet congestion control. In *International conference on machine learning*. PMLR, 3050–3059.

[12] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. 2020. Concept bottleneck models. In *International conference on machine learning*. PMLR, 5338–5348.

[13] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 4765–4774. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[14] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication.* 197–210.

[15] Zili Meng, Minhu Wang, Jiasong Bai, Mingwei Xu, Hongzi Mao, and Hongxin Hu. 2020. Interpreting deep learning-based networking systems. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication.* 154–171.

[16] Sagar Patel, Junyang Zhang, Sangeetha Abdu Jyothi, and Nina Narodytska. 2023. Plume: A Framework for High Performance Deep RL Network Controllers via Prioritized Trace Sampling. *arXiv preprint arXiv:2302.12403* (2023).

[17] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

[18] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *International conference on machine learning*. Pmlr, 8821–8831.

[19] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining.* 1135–1144.

[20] Shivani Singh, Razia Sulthana, Tanvi Shewale, Vinay Chamola, Abderrahim Benslimane, and Biplab Sikdar. 2021. Machine-learning-assisted security and privacy provisioning for edge computing: A survey. *IEEE Internet of Things Journal* 9, 1 (2021), 236–260.

[21] Zhenyu Song, Kevin Chen, Nikhil Sarda, Deniz Altınbüken, Eugene Brevdo, Jimmy Coleman, Xiao Ju, Pawel Jurczyk, Richard Schooler, and Ramki Gummadi. 2023. {HALP}: Heuristic aided learned preference eviction policy for {YouTube} content delivery network. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23).* 1149–1163.

[22] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2020. BOLA: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking* 28, 4 (2020), 1698–1711.

[23] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).

[24] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533* (2022).

[25] Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. 2020. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20).* 495–511.

[26] Francis Y Yan, Jestin Ma, Greg D Hill, Deepti Raghavan, Riad S Wahby, Philip Levis, and Keith Winstein. 2018. Pantheon: the training ground for Internet congestion-control research. In *2018 USENIX Annual Technical Conference (USENIX ATC 18).* 731–743.

[27] Yue Yang, Artemis Panagopoulou, Shenghao Zhou, Daniel Jin, Chris Callison-Burch, and Mark Yatskar. 2023. Language in a bottle: Language model guided concept bottlenecks for interpretable image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 19187–19197.

[28] Mert Yuksekgonul, Maggie Wang, and James Zou. 2022. Post-hoc concept bottleneck models. *arXiv preprint arXiv:2205.15480* (2022).

[29] Yanqi Zhang, Weizhe Hua, Zhuangzhuang Zhou, G Edward Suh, and Christina Delimitrou. 2021. Sinan: ML-based and QoS-aware resource management for cloud microservices. In *Proceedings of the 26th ACM international conference on architectural support for programming languages and operating systems.* 167–181.

[30] Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 67, 2 (2005), 301–320.