Agua: A Concept-Based Explainer for Learning-Enabled Systems

Sagar Patel
University of California, Irvine

Dongsu Han
KAIST

Nina Narodystka
VMware Research by Broadcom

Sangeetha Abdu Jyothi
University of California, Irvine

ABSTRACT

While deep learning offers superior performance in systems and networking, adoption is often hindered by difficulties in understanding and debugging. Explainability aims to bridge this gap by providing insight into the model's decisions. However, existing methods primarily identify the most influential input features, forcing operators to perform extensive manual analysis of low-level signals (e.g., buffer $t-1$).

In this paper, we introduce Agua, an explainability framework that explains a model's decisions using high-level, human understandable *concepts* (e.g., "volatile network conditions"). Our concept-based explainability framework lays the foundation for intelligent networked systems, enabling operators to interact with data-driven systems. To explain the controller's outputs using concept-level reasoning, Agua builds a surrogate concept-based model of the controller with two mappings: one from the controller's embeddings to a predefined concept space, and another from the concept space to the controller's output. Through comprehensive evaluations on diverse applications—adaptive bitrate streaming, congestion control, and distributed denial of service detection—we demonstrate Agua's ability to generate robust, high-fidelity (93-99%) explanations, outperforming prior methods. Finally, we demonstrate several practical use cases of Agua in networking environments—debugging unintended behaviors, identifying distribution shifts, devising concept-based strategies for efficient retraining, and augmenting environment-specific datasets.

CCS CONCEPTS

• **Networks** → *Application layer protocols*; • **Computing methodologies** → **Control methods**; **Neural networks**;

KEYWORDS

Machine Learning for Systems, Explainability, Large Language Models

ACM Reference Format:
Sagar Patel, Dongsu Han, Nina Narodystka, and Sangeetha Abdu Jyothi. 2025. Agua: A Concept-Based Explainer for Learning-Enabled Systems. In *ACM SIGCOMM 2025 Conference (SIGCOMM '25), September 8–11, 2025, Coimbra, Portugal.* ACM, Coimbra, Portugal, 18 pages. https://doi.org/10.1145/3718958.3754341

SIGCOMM '25, September 8–11, 2025, Coimbra, Portugal

ACM ISBN 979-8-4007-1524-2/2025/09.
https://doi.org/10.1145/3718958.3754341

1 INTRODUCTION

Today, learning-enabled controllers outperform manually designed ones in a range of computer systems and networking applications, such as adaptive bitrate streaming [36, 70], congestion control [6, 24, 71], resource management [79], edge caching [60], or network security [59]. However, operators often struggle with these solutions because they are difficult to interpret, debug, and reason about [38].

Current explainability solutions for learning-enabled controllers focus primarily on interpreting their decisions by identifying the most influential input features. For example, Metis [38] 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 [23] identifies whether the learned model contains any inductive bias by analyzing the decision tree.

However, these feature-based solutions have intrinsic limitations. They typically involve large decision trees with hundreds of nodes and complex decision paths. For example, the video streaming explainer generated using Trustee in Fig. 1c has the 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 explanations, while more meaningful than a neural network, are difficult to understand as they involve dozens of decision points across disparate features. More importantly, existing feature-based explanation techniques [33, 54] fail to account for high-level concepts that emerge from interactions among multiple features. For instance, identifying volatile network conditions often cannot be attributed to a single feature, but instead requires analyzing patterns across multiple features and time instances to understand their combined impact.

Acknowledging these fundamental limitations of feature-based techniques, a new generation of interpretable models was introduced in computer vision: concept-bottleneck models (CBMs) [26, 72, 74]. Unlike traditional methods that rely on low-level features, these models first predict higher-level concepts (e.g., 'whiskers' or 'tail') and then use them to obtain the final output (e.g., 'cat'). This approach enables users to reason intuitively about the model's decisions without needing to inspect individual pixel values.

We argue that concepts can similarly have the potential to bridge the gap between operators and intelligent systems. In systems settings, we envision *concepts* as high-level abstractions of controller behaviors that span multiple timestamps and features, capturing patterns that simple feature-based explanations overlook. For example, in video streaming, concepts such as 'Anticipation of congestion' or 'Rapidly depleting buffer' offer clearer insights than raw values of throughput or delay. This improved clarity can enhance transparency and operational control, allowing operators to effectively manage intelligent data-driven systems.

However, concept-based techniques used in computer vision [18, 20, 26, 72, 74] cannot be directly applied to systems environments due to several reasons: (i) systems controllers take as inputs heterogeneous unlabeled data, while vision models handle images, with wide availability of labels. (ii) Concepts in learning-enabled systems are more complex, usually requiring multiple levels of abstraction. (iii) Systems controllers typically handle complex temporal dynamics and long-term dependencies. Thus, while we draw inspiration from the notion of *concepts* in the computer vision domain, prior techniques are not directly applicable to systems settings. Consequently, a new approach to concept-based understanding is required in learning-enabled systems.

In this paper, we reimagine concept-based explainability for systems and introduce Agua, the first framework for it. To address the unique challenges of system environments, we introduce a concept generation mechanism combining domain knowledge, data-driven analysis, and Large Language Models (LLMs) [7, 12, 17, 65]. First, we provide relevant literature or design documents to an LLM and derive a set of base concepts. These base concepts may be further augmented or filtered by the operator. Next, we develop a surrogate concept-based model of the controller, which learns a mapping from the input space to the output space of the controller, with the base concepts as an intermediate representation. We achieve this by splitting the surrogate model into two parts. We first build the concept mapping function, which transforms controller inputs into a space of these base concepts. Then, we learn the output mapping function, which linearly combines those concepts to reconstruct the controller's output. This two-stage surrogate model enables Agua to reveal the main concepts driving its behavior. It is also broadly applicable to heterogeneous systems with supervised, semi-supervised, or reinforcement learning controllers.

We demonstrate Agua's effectiveness in three different applications: adaptive bitrate streaming (ABR) [51], congestion control (CC) [24], and DDoS detection [16]. In addition to being easy to understand, Agua achieves high fidelity, exceeding 0.933 in all applications, and outperforms Trustee by up to 0.69. Agua unlocks new capabilities at each stage of the system lifecycle. In design and testing, Agua highlights which concepts underlie unexpected behaviors, enabling operators to intuitively diagnose and fix them. In deployment, it detects shifts in the distribution by monitoring shifts in key concepts over time. In retraining, Agua leverages concept-level insights to guide model updates and data augmentation, improving performance.

In summary, we make the following contributions:

- We propose concepts as high-level abstractions spanning multiple features and timestamps and capturing complex patterns to understand learning-enabled systems.
- We put forward Agua as the first concept-based explainer for systems, combining the power of domain knowledge, data-driven analysis, and LLMs.
- We evaluate Agua on adaptive bitrate streaming, congestion control, and DDoS detection to show Agua's high fidelity and low complexity, outperforming prior state of the art.
- We demonstrate the new capabilities that Agua unlocks at each stage of the system lifecycle, identifying and debugging unintended behavior, detecting distribution shifts, guiding retraining, and augmenting workload datasets.
- We validate the robustness of Agua, withstanding noise at multiple points in its pipeline, including variations in LLM outputs and across both open- and closed-source LLMs.

This work does not raise any ethical concerns.

2 RELATED WORK AND MOTIVATION

Explainers shed light on the reasoning behind an opaque machine learning model. We begin by providing a brief overview of the prior techniques. Then, we highlight the limitations of these techniques and discuss the opportunities presented by concepts in the computer vision domain. Finally, we detail the roadblocks in adopting them in the context of systems and motivate the need for a tailored solution.

2.1 Prior Explainers

Prior explainability approaches in the systems domainn predominantly identify the most important input features—we categorize this line of work as "feature-based explainability". Feature-based explainers can be classified into two categories based on their scope: (i) local explainers and (ii) global explainers.

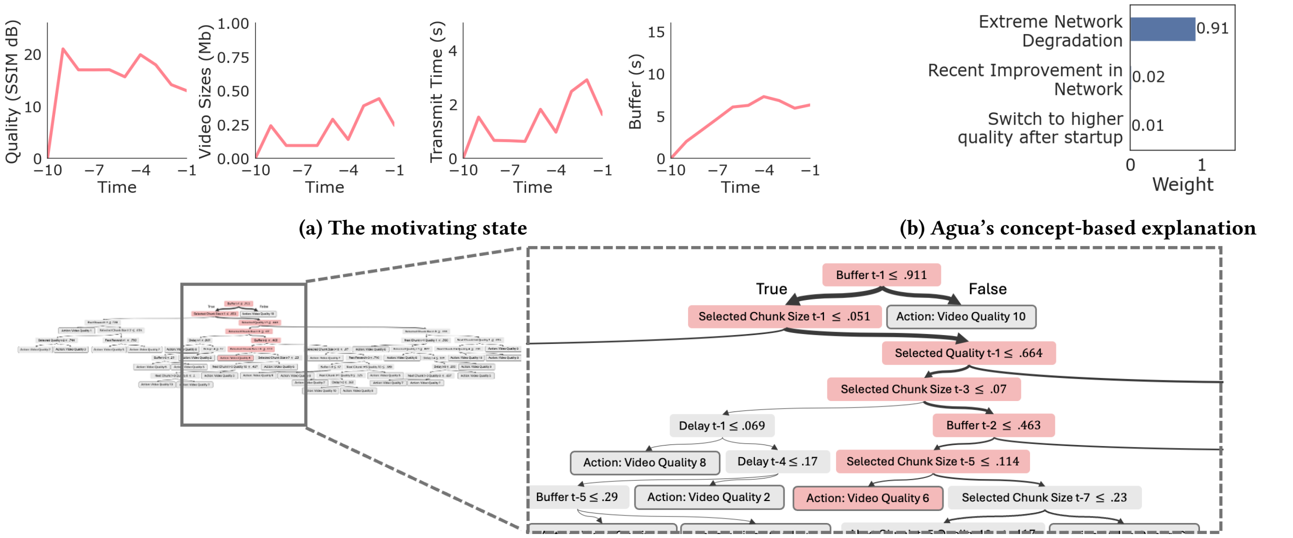
(i) Local Feature Explainers. Local explainers, such as LIME [54], SHAP [34], and Captum [28], explain the model's prediction for a specific input instance. They generate perturbed samples by introducing small changes to the input and observing the model's predictions on these samples. A surrogate model, such as a linear regression [34, 54] or a saliency map [40, 62], is then trained on the perturbed dataset to approximate the decision logic. The parameters of the surrogate model reveal the top features for the prediction.

(ii) Global Feature Explainers. Global explainers approximate the behavior of the entire model across the input space. They train an interpretable surrogate model, such as a decision tree [63, 76], graph [75, 77], or rule set [42], to mimic the outputs for a dataset of representative inputs. In the systems domain, Metis [38] constructs global decision trees and hypergraphs to approximate neural network behavior. Trustee [23] builds upon Metis by better balancing fidelity, complexity, and stability, alongside providing a trust report.

2.2 Shortcomings of Prior Explainers

We demonstrate the limitations of feature-based explainers using Trustee [23] with the state-of-the-art adaptive bitrate (ABR) controller, Gelato [51]. Gelato is a deep RL controller, which takes as input the history of the client's viewing experience and gives as output the next bitrate.

Motivating Scenario. Consider the scenario where the operator seeks to understand why Gelato picks a low bitrate despite a recovering buffer in a particular state (Figure 1a). Trustee [23] 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 has 61 nodes and a depth of 10. We generate an explanation for the motivating scenario by traversing the highlighted decision path on the pruned tree (Fig. 1c). Despite this, the explanation is difficult to understand, involving seven decision nodes on multiple features: buffer, selected chunk size, and upcoming video qualities—split across time.



(a) The motivating state

(b) Agua's concept-based explanation

(c) The decision path of the state in the pruned Trustee [23] report

Figure 1: Trustee [23]'s explanation for the motivating state. The full and pruned decision trees in its report are both large and even filtering to the decision path for the motivation state leaves a complex explanation.

Feature-level explanations force human operators to understand the intricacies of the feature set originally designed for a machine. This often proves difficult without extensive data science expertise and manual effort. This limitation is inherent to all feature-based explainers [28, 33, 34, 54, 63, 76].

2.3 Concept-level View and Challenges

Recognizing the limitations of feature-based solutions, in the computer vision domain, self-interpretable Concept-Bottleneck Models (CBMs) [26, 72, 74] were proposed. CBMs modify the neural network to introduce an intermediate concept-bottleneck layer. A CBM first takes the image, passes it through the concept bottleneck layer to predict the concepts present in the image, and then combines the concepts to obtain the label for the image. For example, a model may predict "bird" using weights on concepts "beak" and "wings".

Inspired by these approaches, we believe that concepts can similarly act as the bridge between data-driven systems and human understanding. In systems, concepts can represent high-level, well-defined ideas grounded in domain-specific knowledge (e.g., 'increasing packet loss' in congestion control). For instance, consider Figure 1b which shows Agua's explanation for the motivating scenario. Each bar represents how much a concept contributes to the decision probability of the action. It is clear that the controller chose the low-quality bitrate because it detected 'Extreme Network Degradation.' These concept-based insights allow operators to quickly and clearly understand the controller rationale and allow operators access to expert-like, high-level reasoning.

However, applying concept-based techniques from the computer vision domain to systems is non-trivial. In computer vision, a key enabler has been the availability of tools to tag concepts in images. Manual tagging approaches [15, 18, 20, 26] rely on the simplicity of image labeling and use crowdsourcing platforms such as MTurk [1]. More recent techniques [72, 74] employ multimodal text-to-image models such as CLIP [53]. In contrast, systems data is heterogeneous and unlabeled, making defining and labeling concepts difficult. Furthermore, system concepts are more complex, requiring multiple levels of abstraction. Lastly, system controllers face intricate temporal dynamics and long-term dependencies, further complicating reasoning. These factors prevent us from relying on the mechanisms from vision techniques.

Despite this, we argue that concepts can still be the path towards accurate and clear explanations. We therefore propose rethinking concept-based explainability for systems.

3 DESIGN OF AGUA

We introduce Agua[1], the first framework for concept-based explainability in learning-enabled systems. Agua enables operators to understand the controller's behavior using high-level, expert-like concepts (e.g., "volatile network conditions") rather than individual low-level features as in prior explainers [23, 34, 54]. This change in perspective aligns the explanations with the expertise and needs of the operators [48].

Agua achieves this by building a surrogate concept-based model of the controller, with concepts as an intermediate representation. This surrogate model has two key components: a concept mapping function that translates the controller's inputs into a concept space, and an output mapping function that converts concepts back to the controller's output. Agua can reveal the linear combination of concepts driving it, giving operators a direct view of the controller's behavior.

Agua builds this surrogate model through a multi-stage training pipeline integrating domain knowledge, data-driven analysis, and LLMs (Figure 2). ❶*Base concept generation*: We generate the set of

[1] https://github.com/NetSAIL-UCI/agua

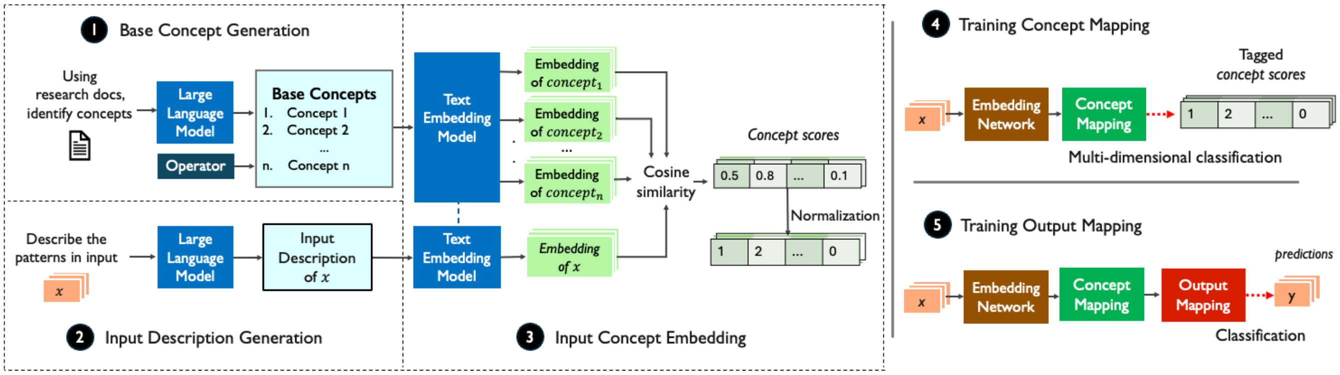


Figure 2: **Training workflow. ❶ We derive the base concepts. Then, ❷ we generate descriptions of inputs using an LLM. Next, ❸ we embed the base concepts and input descriptions with a text embedding model and get concept similarity scores. ❹ We then sequentially train the concept mapping function using the previously generated concept similarity scores (represented as a vector with concepts as rows and the similarity score as the integer class). ❺ Finally, we train the output mapping function with the corresponding controller outputs y serving as our target.**

base concepts from domain literature using an LLM, which may be further filtered or augmented by an operator. We then gather training data for the surrogate model in two stages. ❷*Input description generation*: We take each input from the dataset and prompt an LLM for a structured *text* description. ❸*Input Concept Embedding*: We embed the base concepts and the previously generated input descriptions using a text embedding model and empirically tag *concept similarities.* ❹*Training Concept mapping*: We use the tagged concept similarities and train a concept mapping function that transforms the controller's embeddings into the concept space. ❺*Training Output mapping*: We train the output mapping function, transforming the concept space back into the controller's output.

After training, Agua can explain the original controller's decision-making for a given input by doing a forward pass through its surrogate model. The weights on the linear combination of concept scores in the output mapping function indicate the top concepts.

3.1 Definitions

Formally, we define the problem of training a concept-based explainer under the standard explainability view, aiming to find the best surrogate approximation of a given model f.

Definition 3.1 (Explainer). An explainer f' is a surrogate function that approximates the original by minimizing the distance metric E between their outputs:

$$f' = \arg\min_{f'} E\big(f(x), f'(x)\big)$$

where f' maps inputs x to outputs y, and E quantifies the discrepancy between $f(x)$ and $f'(x)$. E also serves as the standard metric for quantitatively evaluating f' (e.g., the fidelity metric in § 4).

Definition 3.2 (Concept-Based Explainer). A concept-based explainer's surrogate function is composed of two functions:

$$f'(x) = \Omega\big(\delta(x)\big) = W^T\delta(x) + b$$

where $\delta : \mathbb{R}^I \to \mathbb{R}^C$ is the concept mapping function that infers concept similarity scores C (e.g., whether 'volatile network conditions' are present) in input x, and $\Omega : \mathbb{R}^C \to \mathbb{R}^n$ is the output mapping function that linearly combines these concept scores to reconstruct the controller's output vector y. The output mapping function serves as the point of explanation in this model—it is a self-interpretable function, with concepts as its input and the controller's output as its output. $\mathbb{R}^I$ is the input space, $\mathbb{R}^C$ is the concept space, and $\mathbb{R}^n$ is the output space. Note that this definition also allows $\delta(x)$ to be a composition of transformations on x. Often, $\delta(x)$ includes parts of the controller's operations on x, making use of available low-dimensional features of x and aligning f' to the controller's space.

Definition 3.3 (Concept-Based Explanation). Given input x and the output y of the model, a concept-based explanation is a vector w that reflects how each base concept contributes to y. It is obtained by backtracking $f'(x)$ through Ω to retrieve the linear combination of concepts that form y.

3.2 Deriving Base Concepts

Base concepts are the building blocks of Agua's explanations, with the controller's output expressed as a weighted combination of these concepts. They must meet the following criteria: (i) they must relate to the controller, (ii) be inferable from its input features, (iii) cover the controller's output space at an appropriate level of abstraction, and (iv) minimally overlap with other concepts.

However, constructing a set that meets all the requirements can be challenging, even for experts. To facilitate this, we design a process to get a starting set by combining the power of survey papers and design documents, empirical analysis, and Large Language Models (LLMs) [7, 12, 65].

We attach a relevant survey paper to the LLM prompt and instruct it to "focus on the background and prior work sections of the paper and summarize factors that influence the controller decision-making in {target domain}." These survey papers contain the research behind the controller and anchor the LLM to relevant retrieved facts. Using this rich context, we repeatedly prompt the LLM to "list and describe the key concepts in the decision y of a controller," iterating through each possible output. We visually describe this process in Figure 2 (❶ 'Concept generation').

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

(a) Concepts for Adaptive Bitrate Streaming

1. Increasing Packet Loss
2. Decreasing Packet Loss
3. Stable Network Conditions
4. Rapidly Increasing Latency
5. Rapidly Decreasing Latency
6. Volatile Network Conditions
7. Low Network Utilization
8. High Network Utilization

(b) Concepts for Congestion Control

1. Geographical and Temporal Consistency
2. Typical Application Behavior
3. Low-and-Slow Attack Indicators
4. High Request Rates
5. Geographic Irregularities
6. Protocol Anomalies
7. Repeated Access Requests
8. Behavioral Anomalies
9. Payload Anomalies
10. Protocol Compliance

(c) Concepts for DDoS Detection

Table 1: Base Concepts used for the applications Adaptive Bitrate Streaming, DDoS Detection, and Congestion Control. These serve as the unit of explanation for Agua, with the controller's output expressed as a linear combination of them.

While informative, because the LLM is not aware of the four criteria we outlined above, this starting set may not meet all of them. The operator may need to use their domain expertise to augment, adjust, or filter it. If there are overlapping concepts, the explanations can be difficult to understand. But if there are not enough concepts, the explanations can be of low fidelity. To help the operator filter, we introduce an empirical check by comparing inter-concept similarities. Since the concepts are rich text descriptions, we can embed them using a text embedding model [4, 66] and create a cosine similarity matrix, where each entry (i, j) represents the similarity between concept C_i and concept C_j:

$$\text{ConceptSimilarity}_{i,j} = \frac{\epsilon(C_i) \cdot \epsilon(C_j)}{\|\epsilon(C_i)\| \|\epsilon(C_j)\|}, \quad \forall i, j \in C. \tag{1}$$

Here, ϵ represents the text embedding function, and C is the set of concepts. We iterate through this matrix and remove any concepts that exceed a similarity threshold $S_{\max}$ with previously retained concepts.

Table 1 lists the base concepts derived by the process for different applications. While DDoS detection (1c) required minimal intervention, adaptive bitrate streaming (1a) required augmenting the starting set, and congestion control (1b) required filtering and adjustments. When a survey document with a different perspective from the controller's design document is used, the set of base concepts can require more tuning. Fidelity could act as a key guide through this process. If the data or concepts are inadequate, they produce low-fidelity explanations. Robustness analysis (such as the ones conducted in Section 5.3 and Appendix A.1) can also help.

3.3 Training Data for Agua's Model

To train Agua's surrogate model, we need labeled data, i.e., we need the controller's inputs labeled with the base concepts present in them. However, as mentioned in Section 2.2, designing a concept labeling mechanism for networked system controllers is challenging. Commonly used methods, such as crowdsourcing or multimodal models, are not feasible due to the heterogeneous nature of inputs in the systems domain. Moreover, since system concepts are inherently complex, a single concept may span multiple input features across time and employ multiple levels of abstraction.

To address these challenges, we prepare the training data in two stages. First, we convert the inputs of system controllers to text descriptions. This step allows us to transform the heterogeneous input space of the controller to a unified semantic space. We automate this process using an LLM by providing the LLM with the base concepts, the input features, and brief descriptions of each feature. We then prompt the LLM to describe the input sample by filling in predefined blanks. For an example of the full prompt and the resulting description, see Figures 15 and 16 in the Appendix. This structured, chain-of-thought prompting [11] guides the LLM to closely examine the input features and how they might be associated with the underlying concepts (❷ 'Input Description Generation')

Second, we empirically measure semantic similarity of the full text descriptions of the inputs to the base concepts. We embed both the base concepts and the input descriptions using a text embedding model [4, 41, 66], and measure the distance between those vectors. Then, we quantize this similarity score into three levels, low, medium, and high; with a concept considered 'absent' if the similarity score is low. By splitting the similarity scores into classes rather than a boolean, Agua enables even low-similarity concepts to be dominant. This can be important in scenarios such as the one given in Fig. 4b where the absence of 'high network throughput' is dominant, and allow Agua to generalize across the multiple levels of abstraction needed in systems applications (❸ 'Input Concept Embedding').

Formally, we compute the cosine similarity between the text embedding, $\epsilon(x)$, of the input description, x, and each base concept, $\epsilon(c_i)$. These similarity scores are then quantized into one of k discrete classes using a quantization function $\psi_k(\cdot)$. S_{c_i}, the similarity class assigned to concept c_i in the set of concepts C, is expressed as:

$$S_{c_i} = \psi_k \left(\frac{\epsilon(x) \cdot \epsilon(c_i)}{\|\epsilon(x)\| \|\epsilon(c_i)\|} \right), \quad \forall c_i \in C. \tag{2}$$

These two stages together allow us to create a structured representation of concepts in systems, avoiding the need for impractical manual labeling or the need to engineer a large multimodal model for systems. For details on the learning process and normalization, please refer to Section 4. We note that this training data does not create explanations themselves: the explanations are obtained by using the trained surrogate model.

3.4 Learning Agua's Surrogate Model

In this section, we present the design of our algorithm to learn Agua's surrogate concept-based model. This surrogate model mimics the original controller model while using concepts as an intermediate representation. With it, we can reveal the contributions of each concept towards the controller's output.

Following Definition 3.2, Agua's surrogate function is a composition of two functions: the concept mapping and the output mapping functions.

❹ **Concept Mapping Function.** The concept mapping function generates a mapping from the controller inputs to the concept space. More specifically, the concept mapping function of Agua takes as input the controller's embeddings of input x, denoted as $h(x)$. This allows Agua to use the available low-dimensional features of x in the embedding space and align itself to the controller.

The concept mapping function δ_θ is formulated as:

$$\delta_\theta : h(x) \rightarrow \mathbb{R}^{C \cdot k}, \quad h(x) \in \mathbb{R}^H, \tag{3}$$

Here, h represents the controller's embedding network, and x is the input. The embeddings are dense vectors of dimension H, determined by the controller's architecture. δ_θ is a multi-label classification model that outputs a vector of size $C \cdot k$ where C is the number of concepts and k is the number of similarity levels tracked. If we rearrange this vector to a matrix of size $C \times k$, each element (i, j) would represent the probability of concept C_i having similarity k_j.

Training the Concept Mapping Function: We frame the training of the concept mapping function δ_θ as a multi-label classification problem and train δ_θ to minimize the cross entropy-loss function:

$$l(x, S_C) = \frac{1}{C} \sum_{i=1}^{C} \left[-\log \left(\frac{e^{\delta_\theta(h(x))_{c_i, S_{C_i}}}}{\sum_{j=1}^{k} e^{\delta_\theta(h(x))_{c_i,j}}} \right) \right] \tag{4}$$

where S_C denotes the ground truth similaties and the element S_{C_i} is the similarity level of concept C_i (e.g., similarity level *high* of concept 'stable network conditions'). The numerator captures the model's score for true similarity S_{C_i} of concept C_i, and the denominator represents the softmax normalization. Note that during the training phase, gradients are not propagated back to the controller's parameters, ensuring that the original neural network, including the controller's embedding network, remains unchanged.

❺ **Output Mapping Function.** The output mapping function Ω (Definition 3.2) maps from the output of the concept mapping function, δ_θ, to the output space of the controller, y. We define Ω as the *linear* function:

$$\Omega(\delta_\theta(h(x))) : y = W^\top \delta_\theta(h(x)) + b \tag{5}$$

where W is the weight matrix, b is the bias vector and y is the output: a vector of n dimension. For classification controllers, n represents the number of output classes, with each element providing the probability of the class. For regression controllers, n corresponds to the dimensionality of the discrete bins used to approximate the numerical output. In this case, the dot product $\Omega(\delta_\theta(h(x))) \cdot bins$ gives the numerical output.

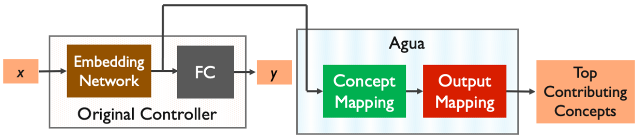


Figure 3: Explanation workflow. After training, Agua takes the controller embedding of an input and provides the linear combination of concept weights for the output.

Training the Output Mapping Function: We employ supervised learning to train the output mapping function Ω, using mini-batch stochastic gradient descent to minimize deviations from the controller. While this training can produce an accurate output mapping function, it introduces a fidelity-complexity trade-off. Achieving high fidelity could require relying on a large number of concepts, which can make the explanations less understandable. To balance this trade-off, we apply ElasticNet regularization [80] to W and b:

$$l_{\text{elastic}} = (1 - \alpha)\|W\|_2^2 + \alpha(\|W\|_1 + \|b\|_1) \tag{6}$$

This regularization encourages sparsity and reduces overfitting by combining both L1 and L2 penalties.

3.5 Generating Explanations with Agua

To explain the output for a given input x, Agua uses its surrogate model to expose the linear combination of concept weights that is behind the output. It does this in three steps. First, it passes x through the controller's embedding network. Second, it applies its concept mapping function to compute the probabilities of concept similarity classes. Third, it uses the output mapping function to extract the linear weights and compute each concept's contribution to the output. Note that this explanation generation process does *not* involve an LLM. LLMs are only used to train the surrogate model of Agua.

We formally derive this explanation using Agua's surrogate model as shown below. In the surrogate model, the output for each individual class i can be computed using the standard dot product:

$$\Omega(\delta_\theta(h(x)))_i = W^{\langle i \rangle} \cdot \delta_\theta(h(x)) + b_i, \forall i \in [1, n] \tag{7}$$

Here, $W^{\langle i \rangle}$ is the i-th row of the weight matrix W, representing the weights for output class i, and b_i is the scalar bias term for it. n here represents the dimension of the output, i.e., the number of potential values that the output can take. This dot product sums over the weighted contribution of all concept similarities and obtains a specific output i.

To extract each concept's contribution, we can rewrite the dot product with a Hadamard product:

$$\Omega(\delta_\theta(h(x)))_i = \|W^{\langle i \rangle} \circ \delta_\theta(h(x)) + \frac{b_i}{C \cdot k}\|_1, \forall i \in [1, n] \tag{8}$$

Here, $W^{\langle i \rangle}$ is applied element-wise (using the Hadamard product $\circ$) to the vector of concept similarities $\delta_\theta(h(x))$, and the L_1 norm sums the resulting vector. If we stop before applying the L_1 norm, we can see exactly how each concept contributes to the output i before being summed. In other words, we can reveal each concept's impact on the output and thus obtain our concept-based explanation.

3.6 Supported Classes of Explanations

Agua supports multiple explanation types with varying scopes.

Factual Explanations. Factual explanations show why the controller chose its output y_i for a given input x. We query Agua for the output class and retrieve the concept weight vector containing the contribution of each concept to the selected output, summing to the result of Ω. However, since Ω's output is not normalized, the weights may not sum to valid probabilities. To allow the concept weight vector to sum to the probability p of the controller output and to intuitively read and rank the top concepts as shown in Fig. 4a, we normalize the concept weights using softmax:

$$\text{concept weight}_i = p(\text{argument-maximum}(y))\sigma(\vec{z})_i \tag{9}$$

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{C.k} e^{z_j}}, \forall i \in C \tag{10}$$

Counterfactual Explanations. Counterfactual explanations provide insight into an output y'_i not chosen by the controller. To enable this, we use Agua's support for arbitrary output class and query for y'_i. With this functionality, operators can understand the conditions under which y'_i could have been selected and what prevented it.

Single Input Explanations. In single input explanations, the operator has an instance x (e.g., from logs or a sample) and requests an explanation for it. In this case, Agua can process x and provide a concept-level breakdown for it.

Batched Input Explanations. Agua can also generalize to batches of inputs. When a batch of inputs is provided, Agua averages the concept contributions over that batch, providing a more holistic view of the controller's behavior.

4 EXPERIMENTAL SETUP

Input Description Generation. We convert the numerical inputs of the controller to the concept space in two stages: (i) input description generation, where we convert the controller inputs to text, and (ii) input concept embedding, where we embed the text descriptions and concepts using a text embedding model and measure their cosine similarity.

In the input description generation stage, our only goal is to use an LLM to transform the inputs to text space. Thus, we want our LLM responses to be as factual as possible: we do not want the LLM to reason about an explanation or add any additional information about these inputs. To enable this, we use standard CoT guidelines [11]. We grounded the prompt by retrieving the relevant features, their descriptions, and the concepts with which we measure similarity. To ensure that the LLM does not deviate from this strict process, we follow the instructions-following guidelines [44] for our LLM models, giving the LLM specific blanks to fill in to describe the state fully. An example of this prompt and the resulting LLM description can be seen in the Appendix in Figure 15 and Figure 16 respectively.

Concept Mapping Function. The concept mapping function, a parameterized function δ of learnable weights θ acts as the bridge between the controller's embedding space and the intermediate concept-space. It is a multi-layer function made up of (i) a Linear layer, (ii) a ReLU activation, (iii) a Layer-Normalization, and (iv) a final Linear layer. Together, these four components transform the controller's embedding space h to the concept similarity space, S_C. The first Linear and its following non-linear activation allow the concept mapping function to learn meaningful information from the controller embeddings. Then, the normalization layer allows this information to shift away from the distribution of the controller embeddings and instead align closer to a normal distribution. This renormalization then allows the final linear layer to accurately predict the concept similarity scores.

Output Mapping Function. The output mapping function converts the concept space back to the controller's output space y. This is done with a single linear layer, which incorporates both a W weight matrix and a b bias vector. This layer-level workflow allows Agua to be natively implemented using standard Deep Learning Libraries and standard learning paradigms.

Training Parameters. In all our experiments, we use the following values for Agua's parameters. For the quantization function used to measure the concept intensities from cosine similarity scores, we use the quantization bins $[0, .2]$, $[.2, .6]$, $[.6, 1.0]$ for the concept similarity classes low, medium, and high, respectively. The concept mapping function is a 2-layer MLP with a LayerNorm [9] in between. In training the concept mapping function, we use batch size 100, learning rate 0.005, and 200 epochs. We additionally use the stochastic gradient descent optimizer with a momentum parameter of 0.25. In training the output mapping function, we use batch size 200, learning rate 0.075 and 500 epochs. We additionally use the elastic regularization parameters α as 0.95 and the regularization coefficient $1e^{-5}$.

Implementation. We implement Agua as a framework in Python. We use the OpenAI API [8] to access GPT-4o-2024-08-06 [2] and the OpenAI large text embedding model [4]. For an open-source variant, we additionally use the HuggingFace library [68] to interface with the Large Language Model LLama 3.3 70B [17] and BAAI BGE-M3 [41] text embedding model. We implement the concept mapping and output mapping functions using Pytorch [46]. We use the Numpy [21] to interface with LLM data and use Scikit-learn [52] for its implementation of the cosine similarity measure.

Fidelity Metric. In explainability, the objective is to understand the predictions of an opaque original model $f(x)$ by approximating it with an interpretable surrogate model $f'(x)$. This surrogate model is designed to mimic the behavior of $f(x)$ while being understandable to humans. Work such as Metis [38], Trustee [23], as well as our explainer, Agua, fall into this explainability category.

The standard way to assess the quality of the surrogate model $f'(x)$ is to measure how well it tracks the original model f [10, 13]. This is done with the fidelity metric, which measures how accurately the surrogate model replicates the original model's predictions over a dataset $\mathcal{D}$.

Application	Trustee [23]		Agua	
	Full	Pruned	LLama 3.3	GPT 4o
ABR	0.946	0.949	0.982	**0.983**
CC	0.215	0.235	0.932	**0.936**
DDoS Detection	0.991	0.977	0.996	**1.000**

Table 2: Explanation Evaluation: We evaluate the fidelity of Agua against the SOTA feature-level explainer, Trustee [23]. We benchmark two versions of Agua, an open-source one with LLM LLama 3.3 70B [17] and BAAI BGE-M3 [41] embeddings, and a closed-source one with LLM GPT-4o [3] and OpenAI large embeddings [4].

$$FD = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(f(X_i) = f'(X_i)), \quad \forall X_i \in \mathcal{D} \tag{11}$$

where n is the number of samples in $\mathcal{D}$, and $\mathbf{1}$ is the indicator function that returns 1 if its argument is true and 0 otherwise. Essentially, the fidelity metric, FD represents the proportion of instances where the surrogate model's predictions match those of the original model. We directly apply this fidelity metric to evaluate our surrogate model and report the results on an unseen test dataset $\mathcal{D}$.

5 EVALUATION

In this section, we first showcase Agua's explanations across three distinct systems applications—adaptive video streaming, congestion control, and DDoS detection. Next, we highlight four practical use cases of Agua that enhance learning-enabled systems, supporting operators throughout the learning systems lifecycle. Finally, we demonstrate the robustness of Agua against noise at various points in its pipeline and analyze its explainability capabilities.

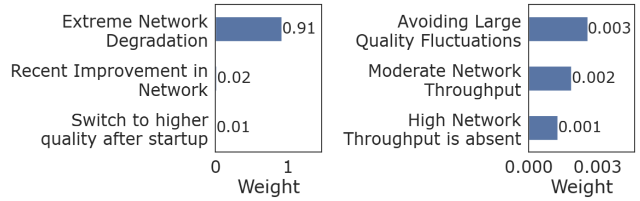
5.1 Agua's Explanations

We apply Agua to three applications and demonstrate the understanding it enables into each.

Adaptive Bitrate Streaming. In adaptive bitrate streaming (ABR), videos are divided into seconds-long chunks and pre-encoded at multiple bitrates. During streaming, the ABR algorithm dynamically selects the quality for each chunk based on network conditions, aiming to maximize the client's quality of experience. Owing to its importance, many learning-enabled controllers have been proposed [36, 51, 61, 70, 78].

We evaluate the state-of-the-art deep RL controller Gelato [51], which handles streaming live television on the Puffer platform [70]. We roll out the controller in its environment and collect 4,000 input-output pairs for training and testing. We then train Agua for the controller, using the base concepts in Table 1a. We additionally generate a Trustee [23] report using the same set (the report in § 2.2).

We observe in Table 2 that Agua successfully captures the controller's behavior, achieving high fidelity and even outperforming Trustee [23]'s reports for the controller. We further use Agua to investigate the decision-making process of Gelato, returning to the motivating example (§ 2.2). We recall that the operator seeks to understand why the controller selected a lower quality bitrate instead of a medium quality option, despite a recovering buffer.



(a) Concept-level explanation for controller's bitrate **(b) Counterfactual explanation for medium quality**

Figure 4: The concept-based explanation for the ABR Motivation Question. The explanations both identify the reason the controller chose the low-quality bitrate and show the top concepts pushing to the medium-quality one.

We first query Agua for a factual explanation for the controller's chosen bitrate in Fig. 4a. The operator can immediately see that the controller selected the low-quality bitrate primarily due to its perception of 'Extreme Network Degradation,' with a minor impact from 'Recent Improvement', potentially latching on to the trends in transmission time of the video chunks. We analyze the input data to verify the concepts. We observe that transmission times, which capture the time taken by the client to receive each 2-second video segment, increased significantly from 1s to 3s in recent history (Figure 1a), confirming significant network degradation. We also observe that it improves to 2s in the most recent timestep, confirming the presence of the concept 'Recent Improvement'.

We further generate a counterfactual explanation for the operator's preferred bitrate, the medium quality, in Fig. 4b. This explanation reveals that the controller would select the medium bitrate if the concepts 'Avoid Large Quality Fluctuations' and 'Moderate Network Throughput' were more dominant and 'High Network Throughput' was absent. The operator was expecting a medium bitrate due to a recovering buffer. However, as Agua reveals, the model's decision-making process gives a higher weightage to throughput-related concepts over buffer-related ones, echoing its reasons for selecting its original medium quality bitrate. Although the buffer has improved, the controller continues to choose a low bitrate because the throughput has not recovered sufficiently. Thus, the factual and counterfactual explanations together provide clear answers to the operator's questions.

Congestion Control. A congestion control (CC) algorithm determines the appropriate data transmission rate over a shared network, aiming to maximize performance while preventing network collapse. Recently, learning-enabled congestion control algorithms have been proposed [6, 24, 71], which rely on complex analyses of network round-trip time, throughput, packet loss rates, etc.

We evaluate Aurora [24], a deep RL controller for CC. Aurora takes as input statistics about the client's latency, throughput, and loss, and outputs a discretized adjustment to the current data transmission rate (from $\frac{1}{2}\times$ to 2×). We roll out the controller in its environment and collect 2,000 training and 4,000 testing input-output pairs. We train Agua for Aurora, reporting the base concepts

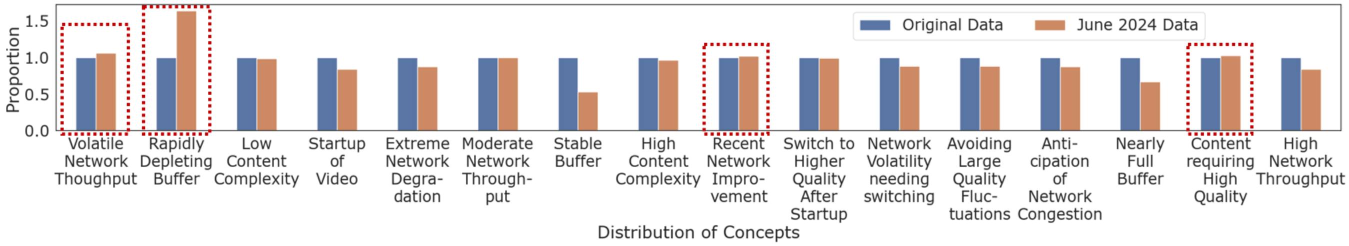
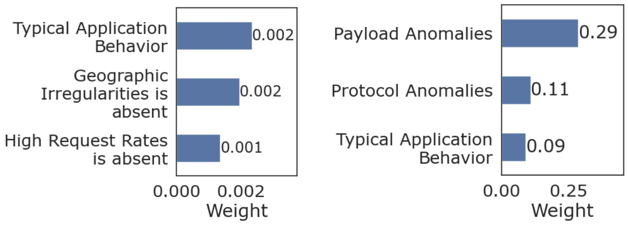


Figure 5: Detecting Datashift in Adaptive Bitrate Streaming deployment with Agua. We analyze the dominant concepts of the inputs in both data and aggregate them. We visualize the resulting normalized proportions.



(a) Explanation for LUCID's prediction of benign traffic

(b) Explanation for flagging of TCP SYN Flag Attacks

Figure 6: Agua's explanation for LUCID's decision making process, revealing its detection mechanism for DDoS flows.

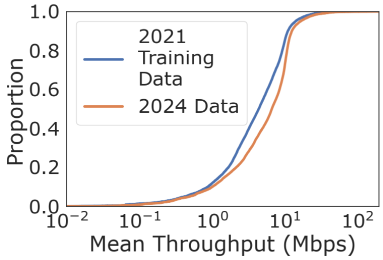


Figure 7: Drift in network throughput experienced by the ABR controller from its 2021 training data and 2024 deployment data.

found in Table 1b. We also generate a Trustee report using the same training and testing set and report the results.

We again observe Agua's high-fidelity in the application in Table 2, significantly outperforming Trustee. This gap highlights the strict trade-offs between fidelity and complexity inherent in feature-level explainers, where providing high-quality explanations often requires a substantial number of individual features. In contrast, Agua generalizes to a higher level and enables operators to understand the controller's behavior in a way natural to them. In this application, the operator notices that the controller produces wide throughput fluctuations under seemingly stable network conditions.

However, due to the fine granularity of congestion control where adjustments are made on millisecond or even submillisecond timescales, understanding the operator's scenario can be difficult using individual input-output pairs. To address this challenge, we leverage Agua's batched inference mode to aggregate insights over time. We roll out the controller under cross-traffic patterns, record its inputs, identify key intervals, and query Agua for factual explanations. Figure 9 visualizes these explanations alongside the throughput, tagging dominant concepts identified by Agua. The shaded region indicates available capacity. From this view, the operator sees that the controller maintains stable throughput when there is no sign of 'volatile network conditions,' but sharply reduces throughput in response to 'rapidly increasing latency', and recovers with 'decreasing packet loss.' This bird's eye perspective shows the reasons behind the throughput fluctuations, which are often missed when focusing solely on individual input samples and rate changes.

DDoS Detection. Distributed Denial-of-Service (DDoS) attacks are among the most disruptive cyberattacks [55]. Attackers often exploit vulnerabilities in IoT devices—such as cameras or thermometers—to assemble botnets that launch volumetric attacks against a single target. Attacks can generate over 1Tbps of traffic [25, 29] and are difficult to detect.

LUCID [16] is a deep supervised learning controller that aims to detect attacks in the brief window between attack initiation and service denial. It takes as input low-level statistics about the flow and predicts whether it is benign or part of a DDoS attack. We follow LUCID's pipeline for the CIC-DDoS2019 dataset [57] and use 1,000 input-output samples for training and 450 for testing. Again, we train Agua for LUCID, and present the base concepts generated in Table 1c.

As with the other applications, we observe that Agua captures the controller with high-fidelity in Table 2, outperforming Trustee. We investigate the mechanism through which LUCID identifies DDoS attacks. First, we evaluate benign flows on LUCID and observe that it correctly classifies them as benign. We query Agua and generate a batched factual explanation for these flows. We visualize this explanation in Fig. 6a. We can observe that the benign flows are primarily identified through 'Typical Application Behavior' and the absence of 'Payload Anomalies', which may be observed through the typical TCP acknowledgment patterns for the HTTP requests found in these flows. To dig further, we evaluate LUCID on flows carrying out TCP SYN Flood attacks and query Agua to understand how it correctly detects these attacks. From Fig. 6b, we can observe that these flows are flagged as DDoS largely due to 'Payload Anomalies' and 'Protocol Anomalies'. These patterns can be evident from the large volume of TCP SYN packets in these flows, where the attacker does not acknowledge the SYN/ACK handshake packets sent by the server, and instead floods the server with more SYN packets. Together, these explanations allow the operator to understand the mechanism through which LUCID detects the attack patterns in the CIC-DDoS2019 workflows and highlight LUCID's ability to do so consistently.

5.2 Applying Agua to enhance systems

In this section, we demonstrate the practical capabilities that Agua unlocks in learning-enabled systems. The abstraction of concepts enables reasoning about the data-driven system using domain-specific terminology familiar to operators and enables them to manage each stage of the system lifecycle—from design and testing to deployment and retraining.

5.2.1 ***Concept-Based Distribution Shift Detection***. Learning-enabled systems often interact with dynamic workloads or real-world network conditions during the course of deployment. In such settings, data distribution shift is common. For example, during the more than 1-year deployment cycle of the ABR controller Gelato, we can witness indications of a shift in the network conditions of clients. The standard approach to analyze this is to measure the two distributions around a key variable [27]. For ABR, this is commonly the client throughput. Thus, in Fig. 7, we plot the throughput distribution from the Puffer platform data collected in June 2024 and the training collected in April-May 2021. We notice that the distribution has changed considerably but do not get an understanding of the nature of the shift.

Agua's concept-based perspective of the controller can solve this problem. We rollout the controller in both datasets and obtain the states the controller encounters within all the traces. We aggregate the states at a trace level and generate Agua's batched input explanations for them. We then tag the traces with the top three identified concepts and plot the proportions of the concepts in both datasets. Figure 5 shows the result. We observe how the distribution has changed at the concept level; conditions with 'volatile network throughput', 'rapidly depleting buffer', 'recent network improvement', and 'high complexity content' have increased while 'stable buffer', 'extreme network degradation', etc have decreased. At this level, operators can precisely judge the impact of the data drift and even define mitigation strategies, as we will see next.

5.2.2 ***Concept-Driven Retraining***. Input-driven deep RL controllers in systems are typically trained using traces from the target environment. In the event of a distribution shift, the data used to train the controller may become stale and hurt performance. In this scenario, a common solution is to retrain the controller over the new data, often in a continuous retraining procedure [70]. However, this process can be tedious and computation-intensive, especially for deep RL controllers in systems [35, 37].

In contrast, Agua's ability to identify how the distribution has shifted at the concept level allows us to design an efficient corrective strategy that addresses the shift without needing to retrain on the entire dataset. More specifically, with Agua's concept tagging of the traces, we can compare the distribution of concepts across the two datasets and identify the ones whose proportions have increased. For example, Fig. 5 shows the concept-level variations across the two Puffer datasets discussed in § 5.2.1. The operator can then retrain on the subset of traces that were insufficiently represented in the original dataset (marked in red in Fig. 5).

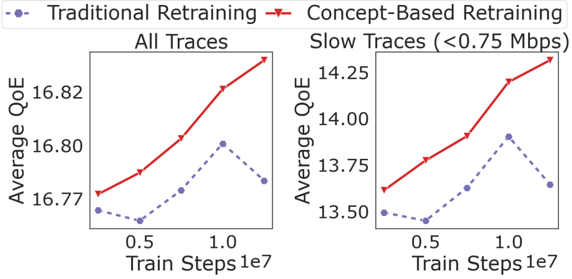


Figure 8: Performance of ABR controller after a retraining procedure targeting the distribution shift.

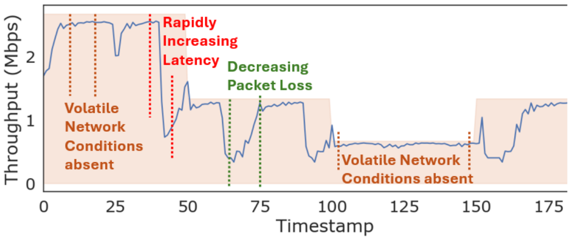


Figure 9: Agua's explanation for Aurora's behavior, with explanations identifying high-level concepts across time.

We compare the result of the concept-driven retraining procedure with traditional retraining on the entire new dataset in Fig. 8. The result shows that concept-based retraining outperforms standard retraining, converging to a higher average QoE in both all and slow network traces. We can additionally observe that the concept-based retraining strategy is stable, steadily improving in QoE across training steps. In contrast, traditional retraining is noisy, gaining and losing QoE throughout the process. Despite training for more than 10 million timesteps, the traditional retraining process does not catch up to the concept-driven approach. This result highlights the prior research demonstrating the difficulty of RL training when the distribution of input traces is wide [37, 51, 69] and also highlights the improvement we can achieve when we focus and filter the distribution with concept-based analysis.

5.2.3 ***Enabling Debuggability***. Debugging learning-enabled systems can be especially challenging because operators must look beyond the current controller and system to uncover hidden failure points. Traditionally, this has meant running repeated statistical analyses, building and interpreting decision trees, and consolidating insights by hand.

With Agua, operators can instead inspect expert-like explanations that unify these steps. As an example, we revisit the congestion control scenario (§ 5.1) where the controller throughput fluctuates despite stable conditions. By examining Agua's explanation (Fig. 9), we can realize that the controller keeps perceiving "rapidly increasing latency" and therefore throttles its throughput too aggressively even when the network conditions are stable.

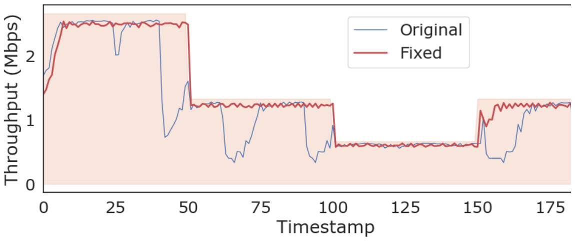


Figure 10: Debugging Aurora with Agua, attaining stable throughput near link capacity.

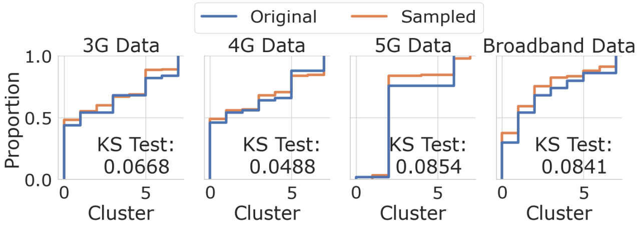


Figure 11: Dataset Expansion, showing that Agua can closely match the target workload distribution with few samples.

Based on this explanation, we can conclude that the controller has a distorted latency perception and thus lacks sufficient latency context. To address this, we add a feature for average latency and extend the history length from 10 to 15. We also adjust training parameters—lowering the learning rate from 1×10^{-4} to 7.5×10^{-5} and increasing entropy—to help the controller converge smoothly on this richer feature set. Figure 10 shows the outcome: the corrected controller (in red) remains steady near full link capacity, while the original (in blue) oscillates. By visualizing the controller’s decisions in Agua, we could naturally identify and implement the changes needed for a stable, high-throughput solution.

5.2.4 *Concept-Guided Dataset Expansion.*

Data is a crucial part of all data-driven learning-enabled systems. However, when working with client workloads or with emerging applications, operators may not always have large datasets. Instead, they may only have access to a large general dataset or a few examples of their target workload. In this case, data expansion or augmentation may be the only choice [58, 64].

With its concept-based perspective, Agua can serve as a powerful tool for dataset expansion. Agua can embed available data in a rich concept space using the data generation workflow (Stages 1, 2, and 3 in Fig. 2) and build a concept-level store of known datasets. Then, with a few instances from a target workload, we can look up the closest samples in the concept space based on cosine similarity and assemble a new larger set similar to the target workload.

To evaluate whether the generated output distribution closely tracks the target workload distribution, we first cluster the available data based on their text embeddings (from Stage 3 in Fig. 2). Each workload typically has a unique distribution of traces across the resulting clusters. For example, the 3G workload in Fig. 11 has 50% traces from cluster 1, 10% from cluster 2, etc. We compare the distribution of the generated and target workloads using the Kolmogorov–Smirnov test (KS test) to measure their similarity. This test calculates the supremum of distances between the empirical CDFs and provides us with a measure of closeness.

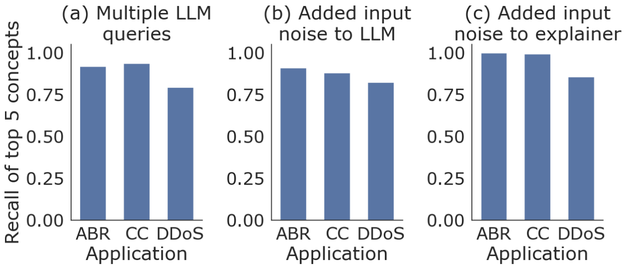


Figure 12: Robustness of Agua: We visualize the impact of faults at multiple points in Agua’s pipeline. In (a), we visualize the robustness of the LLM across multiple queries, in (b) its robustness with more than 5% added noise to the inputs, and in (c) Agua’s explanations across 5% added noise.

We empirically evaluate Agua’s capabilities at this task by assembling four datasets of distinct network workloads: 3G, 4G, 5G, and broadband. We then rollout Gelato on each of these datasets and aggregate the states at the workload level. We then use Agua’s data generation workflow to form a large data store of the workloads. Then, we query this store with a small set of held-out samples of each workload. We assemble a new expanded dataset of each type by aggregating the samples most similar to the target workload in cosine distance. We visualize the result of this analysis in Figure 11, where we plot the empirical CDF of each of the datasets across the unified clustering axis. We observe that the sampled augmented data closely tracks the CDF of the original. We can observe that the KS Test statistic is below 0.08 in each case, signifying that the two distributions are very similar. This demonstrates Agua’s ability to closely track the data and practically alleviate problems throughout the lifecycle of learning-enabled systems.

5.3 Agua Robustness

To further validate the high fidelity of Agua’s explanations, in this section, we conduct a series of experiments to assess the robustness of Agua at multiple points, as shown in Figure 12.

Multiple LLM queries. First, we assess the sensitivity of Agua’s Input Concept Embeddings—values obtained via LLMs and text embedding models—to the inherent randomness in LLM outputs. We repeatedly query the LLM to describe the same input sample, generating an Input Concept Embedding from each description. We then identify the top five highest-intensity concepts across all queries and measure how often these overall top five concepts appear in each individual query’s top five. This recall value is visualized in Figure 12a. We observe that the overall top concepts are recalled over 80% of the time across all applications. This result highlights the robustness of Agua’s data-first empirical approach to concept similarity identification, demonstrating its ability to withstand the inherent randomness in LLM outputs.

Added noise to the LLM input sample. Next, we analyze the robustness of Input Concept Embeddings to noise in the input samples. This evaluation ensures that noisy conditions in data collection (e.g., measurement delays, conversion errors) do not significantly impact the embeddings. We first obtain baseline values for an input sample x by querying for its description and concept mapping. Then, we add noise up to 0.07× the standard deviation of the input to x (about 5% noise) and generate new concept intensities. We collect multiple embeddings from these noisy variations and measure the recall of the baseline top five concepts in each noisy sample's top five. The results are visualized in Figure 12b. We find that Agua's pipeline remains robust to this noise, again achieving more than 0.80 recall across applications, and ensuring the validity of the training data despite potential imperfections.

Added input noise to the explainer. Finally, we analyze the robustness of Agua's fully trained explanations to input sample noise. We generate baseline factual explanations for input samples x, then add the same 5% noise to x and regenerate the explanations. We present the results in Figure 12c, comparing the noisy explanations to the baseline. Due to the stability in the training data, the explainer remains robust, achieving recalls close to 0.9 across applications. These experiments demonstrate the reliability of Agua's explanations against potential failure points.

6 DISCUSSION AND LIMITATIONS

Building upon our prior work [48] on concept-based explainability of learning-based systems, we conceive Agua to be a stepping stone towards concept-level analysis in practical learning-enabled systems rather than a one-stop solution, laying the path for future work.

Adhoc Text Explainability. Besides Agua, there are many unexplored ways to incorporate natural language understanding into data-driven applications—for example, by prompting an LLM to convert decision trees to text, or even by simply having it summarize the controller's behavior. While these directions can be interesting, they can also introduce additional challenges. Solutions in the field of explainability can be difficult to evaluate [43]. To this end, Agua offers a natural evaluation tool, as it is a surrogate model technique similar to Trustee [23] and Metis [38], which can be empirically evaluated through fidelity—a measure of how well it mimics the controller. By contrast, LLM approaches that do not involve building such a model lack such a statistical measure.

Simpler Classes of Machine Learning Controllers. Agua enables understanding deep learning controllers, encompassing a wide range of controllers used in supervised, semi-supervised, and reinforcement learning. However, in its current form, Agua's concept-level view does not generalize to classes of controllers that, while simpler than neural networks, remain challenging to understand (e.g., decision trees, gradient boosting machines, etc). This limitation is due to Agua's reliance on fixed-dimensional vector embeddings from the controller, which may not be obtainable from these simpler classes of models. Research focused on encoding these models [14] could help bridge this gap.

Suboptimal Base Concepts. Base concepts serve as the building blocks of Agua's explanations, with the controller's output represented as a linear combination of these concepts. Since the controller inputs are projected onto this concept space, the quality of the base concepts impacts Agua's fidelity. Similar to decision tree methods, where suboptimal decision basis from its heuristics [31] can lead to lower-fidelity explanations, suboptimal base concepts can reduce explanation fidelity. We observe this in Appendix A.1 when investigating the impact of size of the concept space on fidelity. However, identifying the optimal set of concepts remains a challenging problem [26, 72]. Research for better filtering and augmenting the concept set [56, 72, 73] could further improve Agua's fidelity.

LLM Reliability. LLMs today can be unreliable, hallucinating on the prompt details or producing inconsistent outputs. To address this unreliability, we formulate Agua as a surrogate model explainer. Even though LLMs are used to generate its training data, Agua's explanations rely only on the surrogate model (as seen in § 3.6, LLMs are not used to generate explanations). This separation allows the training process of Agua to withstand the noise of the LLM outputs. The LLM's incorrect description of a controller input does not invalidate its explanation; the explanation is built from training that takes into account the *overall* data. However, this does not insulate Agua from all problems with the LLM. If the LLM *consistently* produces incorrect outputs for all inputs, for example, due to a misbehaving prompt [32] or a bug [45], it can corrupt Agua's training data and produce low-fidelity explanations. Standard checks [67] or validation (e.g., Appendix A.2) to confirm the behavior of LLM can prove vital here. We additionally note that Agua, like Trustee [23] and Metis [38], is a post-hoc explainability method and cannot provide exact guarantees. In cases where multiple pathways can lead to the same prediction, the "explanation" from a post-hoc method may not reflect the actual reasoning of the original model.

Growing LLM Research. The field of AI is evolving rapidly. Open-source LLMs are attaining state-of-the-art performance, and new models are becoming available each day. To incorporate these advances, we evaluate Agua with LLama 3.3-70B [17] in Table 2, finding that this open-source variant performs almost on par with GPT-4o [3]. Since conducting our experiments, OpenAI has launched o1, which ranks among the top at the U.S. Math Olympiad [5], and even unveiled o3, attaining 3× the intelligence of o1 [22]. As these models continue to improve, the steps Agua takes to limit reliance on them may become unnecessary. We see this progression as further motivation to harness the power of language-tied concepts, becoming a crucial step toward the coevolution of AI and computing systems [19, 30, 39].

Unified Explainability. While Agua provides operators with an understanding of the controller's fully trained neural network, it does not provide insights into why it got there. It does not capture the time-dependent control loop: how the controller influences the system and how the system's behavior influences the controller training. However, this perspective can be crucial to designing effective data-driven solutions. With an understanding of how the controller perceives the future, the operator can effectively diagnose not just the resultant controller but also the optimization process used to achieve it. Efforts to combine this future-based perspective [47, 50] with Agua can be a significant step towards this unified perspective on explainability.

Interactive Intent-Driven Controller Design. In light of the rapid evolution of LLMs, we envision that concept-based explanations will enable breakthroughs in data-driven controller design, allowing operators to interact in natural language to design controllers that match their intent. We believe that mapping input states to concepts is a key step toward this agentic AI-enabled design process, as it connects the inputs and outputs of a controller to the complex and nuanced encoding of concepts, semantics, and relationships. By combining the logic and ideas embodied in LLMs with controller-specific data-driven learning, we hope our work paves the path towards a new class of intelligent systems.

7 CONCLUSION

In this paper, we present Agua, a radically new explainability framework designed for human operators based on high-level concepts. We demonstrate Agua's ability to generate high-fidelity and robust explanations, outperforming prior techniques. We further demonstrate Agua's ability to enable operators to intuitively meet key needs throughout the system lifecycle. We envision this work to highlight the potential of high-level concept-based explanations and offer a path towards a framework where operators build and manage intelligent systems through nothing but their domain expertise and familiar language.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful feedback. This work is supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) and National Research Foundation (NRF) of Korea (No. RS-2024-00418784 and RS-2024-00340099).

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.]. Introducing Structured Outputs in the API | OpenAI. https://openai.com/index/introducing-structured-outputs-in-the-api/. (Accessed on 09/18/2024).

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

[5] [n. d.]. OpenAI o1 Hub | OpenAI. https://openai.com/o1/. (Accessed on 09/18/2024).

[6] 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.

[7] 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).

[8] Open AI. [n. d.]. openai-python: The official Python library for the OpenAI API. https://github.com/openai/openai-python. (Accessed on 09/17/2024).

[9] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[10] Vaishak Belle and Ioannis Papantonis. 2021. Principles and practice of explainable machine learning. *Frontiers in big Data* 4 (2021), 688969.

[11] Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. 2023. Prompting is programming: A query language for large language models. *Proceedings of the ACM on Programming Languages* 7, PLDI (2023), 1946–1969.

[12] 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.

[13] Nadia Burkart and Marco F Huber. 2021. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research* 70 (2021), 245–317.

[14] Arthur Choi, Andy Shih, Anchal Goyanka, and Adnan Darwiche. 2020. On symbolically encoding the behavior of random forests. *arXiv preprint arXiv:2007.01493* (2020).

[15] Devleena Das, Sonia Chernova, and Been Kim. 2023. State2explanation: Concept-based explanations to benefit agent learning and user understanding. *Advances in Neural Information Processing Systems* 36 (2023), 67156–67182.

[16] Roberto Doriguzzi-Corin, Stuart Millar, Sandra Scott-Hayward, Jesus Martinez-del Rincon, and Domenico Siracusa. 2020. LUCID: A practical, lightweight deep learning solution for DDoS attack detection. *IEEE Transactions on Network and Service Management* 17, 2 (2020), 876–889.

[17] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[18] Radwa El Shawi. 2024. ConceptGlassbox: Guided Concept-Based Explanation for Deep Neural Networks. *Cognitive Computation* (2024), 1–14.

[19] David B Fogel. 2006. *Evolutionary computation: toward a new philosophy of machine intelligence.* John Wiley & Sons.

[20] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. 2019. Towards automatic concept-based explanations. *Advances in neural information processing systems* 32 (2019).

[21] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[22] Jeremy Hsu. [n. d.]. OpenAI's o3 model aced a test of AI reasoning – but it's still not AGI — newscientist.com. https://www.newscientist.com/article/2462000-openais-o3-model-aced-a-test-of-ai-reasoning-but-its-still-not-agi/. [Accessed 07-01-2025].

[23] 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.

[24] 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.

[25] James A Jerkins. 2017. Motivating a market or regulatory solution to IoT insecurity with the Mirai botnet code. In *2017 IEEE 7th annual computing and communication workshop and conference (CCWC)*. IEEE, 1–5.

[26] 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.

[27] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. 2021. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*. PMLR, 5637–5664.

[28] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, et al. 2020. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896* (2020).

[29] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. 2017. DDoS in the IoT: Mirai and other botnets. *Computer* 50, 7 (2017), 80–84.

[30] Kirill Krinkin, Yulia Shichkina, and Andrey Ignatyev. 2021. Co-evolutionary hybrid intelligence. In *2021 5th Scientific School Dynamics of Complex Networks and their Applications (DCNA)*. IEEE, 112–115.

[31] Hyafil Laurent and Ronald L Rivest. 1976. Constructing optimal binary decision trees is NP-complete. *Information processing letters* 5, 1 (1976), 15–17.

[32] Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, et al. 2023. Prompt Injection attack against LLM-integrated Applications. *arXiv preprint arXiv:2306.05499* (2023).

[33] 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

[34] 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

[35] Hongzi Mao, Shannon Chen, Drew Dimmery, Shaun Singh, Drew Blaisdell, Yuandong Tian, Mohammad Alizadeh, and Eytan Bakshy. 2020. Real-world video adaptation with reinforcement learning. *arXiv preprint arXiv:2008.12858* (2020).

[36] 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.

[37] Hongzi Mao, Shaileshh Bojja Venkatakrishnan, Malte Schwarzkopf, and Mohammad Alizadeh. 2018. Variance reduction for reinforcement learning in input-driven environments. *arXiv preprint arXiv:1807.02264* (2018).

[38] 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.

[39] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. 2021. A graph placement methodology for fast chip design. *Nature* 594, 7862 (2021), 207–212.

[40] Pramod Kaushik Mudrakarta, Ankur Taly, Mukund Sundararajan, and Kedar Dhamdhere. 2018. Did the model understand the question? *arXiv preprint arXiv:1805.05492* (2018).

[41] Multi-Linguality Multi-Functionality Multi-Granularity. [n. d.]. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. ([n. d.]).

[42] W James Murdoch and Arthur Szlam. 2017. Automatic rule extraction from long short term memory networks. *arXiv preprint arXiv:1702.02540* (2017).

[43] Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice Van Keulen, and Christin Seifert. 2023. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. *Comput. Surveys* 55, 13s (2023), 1–42.

[44] OpenAI. [n. d.]. OpenAI Platform — platform.openai.com. https://platform.openai.com/docs/guides/text?api-mode=responses. [Accessed 04-06-2025].

[45] openAI. [n. d.]. Sycophancy in GPT-4o: What happened and what we're doing about it. https://openai.com/index/sycophancy-in-gpt-4o. [Accessed 04-06-2025].

[46] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[47] Sagar Patel, Sangeetha Abdu Jyothi, and Nina Narodytska. 2023. Towards Future-Based Explanations for Deep RL Network Controllers. *SIGMETRICS Perform. Eval. Rev.* 51, 2 (Oct. 2023), 100–102. https://doi.org/10.1145/3626570.3626607

[48] Sagar Patel, Dongsu Han, Nina Narodystka, and Sangeetha Abdu Jyothi. 2024. Toward Trustworthy Learning-Enabled Systems with Concept-Based Explanations. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks.* 60–67.

[49] Sagar Patel, Dongsu Han, Nina Narodystka, and Sangeetha Abdu Jyothi. 2024. Toward Trustworthy Learning-Enabled Systems with Concept-Based Explanations. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks* (Irvine, CA, USA) *(HotNets '24)*. Association for Computing Machinery, New York, NY, USA, 60–67. https://doi.org/10.1145/3696348.3696894

[50] Sagar Patel, Sangeetha Abdu Jyothi, and Nina Narodytska. 2024. CrystalBox: future-based explanations for input-driven deep RL systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 14563–14571.

[51] Sagar Patel, Junyang Zhang, Nina Narodystka, and Sangeetha Abdu Jyothi. 2024. Practically High Performant Neural Adaptive Video Streaming. *Proc. ACM Netw.* 2, CoNEXT4, Article 30 (Nov. 2024), 23 pages. https://doi.org/10.1145/3696401

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[53] 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.

[54] 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.

[55] Mikail Mohammed Salim, Shailendra Rathore, and Jong Hyuk Park. 2020. Distributed denial of service attacks and its defenses in IoT: a survey. *The Journal of Supercomputing* 76 (2020), 5320–5363.

[56] Simon Schrodi, Julian Schur, Max Argus, and Thomas Brox. 2024. Concept Bottleneck Models Without Predefined Concepts. *arXiv preprint arXiv:2407.03921* (2024).

[57] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali A Ghorbani. 2019. Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In *2019 international carnahan conference on security technology (ICCST).* IEEE, 1–8.

[58] Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of big data* 6, 1 (2019), 1–48.

[59] 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.

[60] 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.

[61] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. 2016. CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proceedings of the 2016 ACM SIGCOMM Conference.* 272–285.

[62] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning.* PMLR, 3319–3328.

[63] Sarah Tan, Rich Caruana, Giles Hooker, Paul Koch, and Albert Gordo. 2018. Learning global additive explanations for neural nets using model distillation. *stat* 1050 (2018), 3.

[64] Luke Taylor and Geoff Nitschke. 2018. Improving deep learning with generic data augmentation. In *2018 IEEE symposium series on computational intelligence (SSCI).* IEEE, 1542–1547.

[65] 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).

[66] 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).

[67] Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Venkat Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. 2025. LiveBench: A Challenging, Contamination-Free LLM Benchmark. In *The Thirteenth International Conference on Learning Representations.*

[68] T Wolf. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).

[69] Zhengxu Xia, Yajie Zhou, Francis Y Yan, and Junchen Jiang. 2022. Genet: automatic curriculum generation for learning adaptation in networking. In *Proceedings of the ACM SIGCOMM 2022 Conference.* 397–413.

[70] 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.

[71] 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.

[72] 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.

[73] Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. 2020. On completeness-aware concept-based explanations in deep neural networks. *Advances in neural information processing systems* 33 (2020), 20554–20565.

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

[75] Quanshi Zhang, Ruiming Cao, Feng Shi, Ying Nian Wu, and Song-Chun Zhu. 2018. Interpreting CNN knowledge via an explanatory graph. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[76] Quanshi Zhang, Yu Yang, Haotian Ma, and Ying Nian Wu. 2019. Interpreting cnns via decision trees. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 6261–6270.

[77] Tong Zhang, Fengyuan Ren, Wenxue Cheng, Xiaohui Luo, Ran Shu, and Xiaolan Liu. 2017. Modeling and analyzing the influence of chunk size variation on bitrate adaptation in DASH. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications.* IEEE, 1–9.

[78] Xu Zhang, Yiyang Ou, Siddhartha Sen, and Junchen Jiang. 2021. {SENSEI}: Aligning video streaming quality with dynamic user sensitivity. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21).* 303–320.

[79] 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.

[80] 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.

Appendices are supporting material that has not been peer-reviewed.

A APPENDIX

In this section, we provide additional analysis and design details of Agua. Agua is a surrogate concept-based model that forms an interpretable mapping from the controller's embedding space $h(x)$ to the controller's output space $f(x)$.

A.1 Concept space analysis

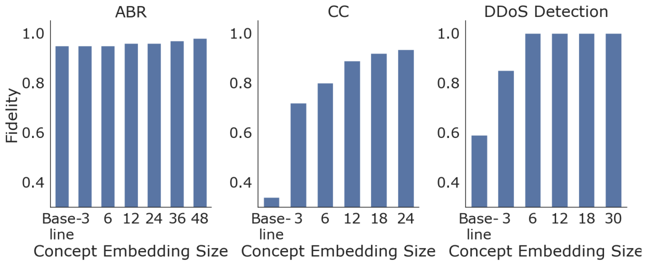


Figure 13: Fidelity across concept size: We visualize the impact of changing the size of the concept space on fidelity. We note that the baseline indicates the fidelity of an explainer that only predicts the most popular output. We see a fidelity-complexity trade-off, with the fidelity improving with larger concept space sizes, but with diminishing returns.

To understand how Agua attains its high fidelity, we investigate the impact of the input concept space size on fidelity. We vary the number of concepts and compare the fidelity against a baseline that always selects the most frequent output. We plot the results in Figure 13. We observe that with a small concept space, the fidelity is low, similar to the baseline. However, as we include more concepts that capture the reasons behind the controller's decisions, the fidelity improves and eventually saturates, showing diminishing returns with larger concept spaces. This experiment illustrates the fidelity-complexity trade-off in concept-based explainability, emphasizing the need to balance fidelity with the number of concepts used while also ensuring high coverage of the controller's output space.

A.2 Text Description Validation

To understand how well the LLM captures human understanding of the input states' patterns and trends, we conduct a small-scale validation experiment. Using ABR as our example domain, we collect a set of controller inputs that cover the output space, obtaining 16 samples. We manually annotate each input, filling in the same prompt given to the LLM. Then, we embed both the LLM and human descriptions using a text embedding model and compute the concept similarity scores for each. We then measure their semantic similarity by computing pairwise differences within them in this concept space. This allows us to understand the impact of the prompts at the concept level. In cosine space, 0 denotes a complete match, while 1 denotes an orthogonal vector. We visualize the distribution of differences in Figure 14. We observe that the differences are small, with more than 80% of the samples having differences under 0.06. We also measure recall of the top 5 concepts, as we did in Section 5.3, and find that it exceeds 0.72. This verifies that the resulting LLM descriptions exhibit a significant semantic overlap with human annotations, validating that the LLM can effectively capture the input dynamics observed by humans at the concept level. In the future, a large scale study involving operators can further investigate this.

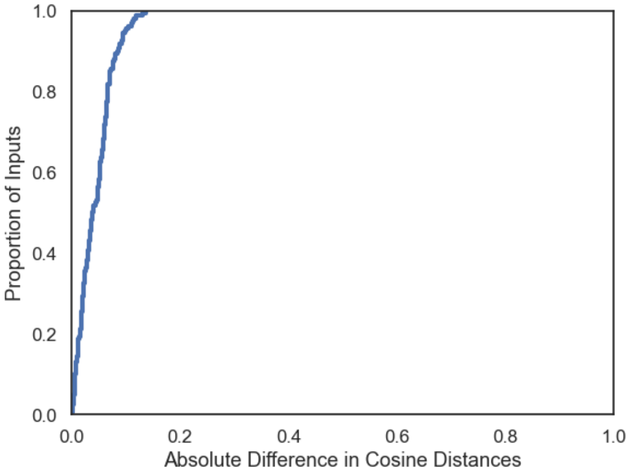


Figure 14: Semantic Similarity of LLM Descriptions: We visualize the semantic similarity of LLM descriptions to human descriptions in the concept space, plotting the distribution of pairwise differences. Note that cosine distance is measured in the range [0, 1], with a difference of 1 indicating complete disjunction.

A.3 Prior Work

We presented the preliminary idea of concept-based explainability for learning-enabled systems in a workshop paper [49]. In the workshop paper, we provide a proof-of-concept implementation for an ABR controller.

In this extended submission, we formalize and implement the concept-based framework Agua. We also formalize the design process and incorporate data-driven filtering and analysis with Agua. We further extensively evaluate the effectiveness of concept-based explainability in three concrete applications and demonstrate the benefit of Agua in four practical use cases in this paper. Lastly, we provide a deep dive into Agua's workflow, assessing its robustness at multiple points and evaluating its concept mapping.

System Instructions: You are a computer scientist trying to gather key information to use in an embedding model to identify patterns. Be straight to the point and avoid unnecessary words.
User Prompt: Explain the patterns in the state using the following key concepts for the environment of Adaptive Bitrate Streaming alongside common statistical metrics. Give an explanation for each takeaway.

Here are the concepts:

(1) **Network Throughput Volatility:** ABR algorithms monitor network throughput. If the throughput significantly decreases, indicating a congested or poor-quality network, the algorithm may choose the lowest quality to ensure continuous playback without buffering.
(2) **Rapidly Depleting Buffer Health:** The health of the video playback buffer is crucial. A rapidly depleting buffer, nearing empty, prompts the algorithm to switch to the lowest quality to quickly refill the buffer and avoid playback interruptions.
(3) **Low Content Complexity:** For content where high resolution is less critical to user experience (e.g., talk shows versus high-action movies), an ABR algorithm may prioritize lower quality streams to conserve bandwidth without significantly impacting perceived quality.
(4) **Initial Playback Start:** To minimize initial loading time, ABR algorithms often start with the lowest quality video and then adapt upwards as more information about network conditions and buffer health becomes available.
(5) **Emergency Fallback:** In scenarios of extreme network degradation or unforeseen technical issues, sending the lowest quality video can serve as an emergency measure to maintain playback continuity until conditions improve.
(6) **Moderate Network Throughput:** Indicates a network condition that, while not optimal, is stable enough to support a video quality above the lowest.
(7) **Buffer Stability with Moderate Occupancy:** The buffer is stable enough to prevent immediate interruptions but not so full as to warrant higher quality streams that might risk future buffering.
(8) **High Content Complexity:** Certain content (fast motion or detailed visuals) may require a higher bitrate to maintain acceptable quality; a slightly higher than lowest quality may be chosen.
(9) **Short-term Network Improvement:** If the algorithm predicts short-term better network conditions, it may select a quality above the lowest to smooth the transition to better conditions.
(10) **Fast Switching Mechanisms from Initial Playback:** To minimize startup delay or adapt quickly, a low but not the lowest quality is chosen as a compromise.
(11) **Adaptive Strategy for Network Fluctuations:** In fluctuating networks, a medium quality compromise avoids the extremes of high and low bitrates.
(12) **Smooth Switching Strategy:** Allows smoother transitions, providing a cushion to adjust down without drastic quality drops.
(13) **Network Congestion Avoidance:** Anticipating congestion, the algorithm may choose a slightly lower bitrate to mitigate rebuffering.
(14) **Robust Buffer Capacity:** A large and stable buffer can handle the data throughput for the highest quality, ensuring smooth playback despite fluctuations.
(15) **Content Suitability:** Content like live sports or visually rich movies benefits significantly from higher quality, justifying a higher bitrate choice.
(16) **Stable, High Network Throughput:** High throughput supports high-quality video streams for a better experience.

State to identify patterns for:

- **Selected Video Quality (SSIM dB), max=25:** [15.250, 15.807, 16.182, 16.998, 16.846, 15.762, 15.371, 15.926, 15.388, 16.080,]
- **Selected Chunk Size (Mb), max=3:** [1.821, 1.949, 2.114, 1.395, 1.208, 1.914, 2.569, 2.291, 2.104, 2.066,]
- **Transmission Time of Chunk (seconds), max=20:** [0.248, 0.160, 0.288, 0.201, 0.177, 0.212, 0.402, 0.320, 0.255, 0.256,]
- **Network Throughput (Mbps), max=3:** [3.000, 3.000, 3.000, 3.000, 3.000, 3.000, 3.000, 3.000, 3.000, 3.000,]
- **Client Buffer (seconds), max=15:** [15.000, 15.000, 15.000, 15.000, 15.000, 14.998, 14.951, 15.000, 14.951, 15.000,]
- **Quality of Experience, max=5:** [3.033, 3.046, 3.115, 3.161, 3.223, 2.974, 3.012, 3.061, 2.996, 3.064,]
- **Stalling (seconds), max=3:** [0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000,]
- **Mean Upcoming Video Qualities (SSIM dB), max=25:** [15.949, 15.480, 14.639, 11.056, 10.676,]
- **Mean Upcoming Video Sizes (Mb), max=3:** [0.272, 0.587, 0.544, 0.891, 0.874,]

Explanation Template:

Network conditions: Initially starts off with (a/an) __ pattern, as observed from the features __. In the middle, it exhibits (a/an) __ to (a/an) __ pattern, as evident from features __. In the end, it exhibits (a/an) __ to (a/an) __ pattern, based on features __. Overall, the trend is __, indicating the presence of __ conditions.

Viewer's video buffer: Initially starts off with (a/an) __ pattern, as observed from the features __. In the middle, it exhibits (a/an) __ to (a/an) __ pattern, as evident from features __. In the end, it exhibits (a/an) __ to (a/an) __ pattern, based on features __. Overall, the trend is __, indicating the presence of __ conditions.

Viewer's Quality of Experience (environment reward or stalling behavior): Initially starts off with (a/an) __ pattern, as observed from the features __. In the middle, it exhibits (a/an) __ to (a/an) __ pattern, as evident from features __. In the end, it exhibits (a/an) __ to (a/an) __ pattern, based on features __. Overall, the trend is __, indicating the presence of __ conditions.

Upcoming video sizes: Initially starts off with (a/an) __ pattern, as observed from the features __. In the middle, it exhibits (a/an) __ to (a/an) __ pattern, as evident from features __. In the end, it exhibits (a/an) __ to (a/an) __ pattern, based on features __. Overall, the trend is __, indicating the presence of __ conditions.

Upcoming video qualities: Initially starts off with (a/an) __ pattern, as observed from the features __. In the middle, it exhibits (a/an) __ to (a/an) __ pattern, as evident from features __. In the end, it exhibits (a/an) __ to (a/an) __ pattern, based on features __. Overall, the trend is __, indicating the presence of __ conditions.

Altogether, the patterns in the features indicate __ network, __ client video player, and __ video stream patterns. This correlates with the key concept of __, __, __, __, and __.

Figure 15: LLM Prompt for an example ABR state, containing the concepts, the state information, and a predefined reasoning pattern for the description.

Network conditions:

- **Initial pattern:** Stable pattern, as observed from the features Network Throughput [3.000 consistently], indicating a constant high availability.
- **Middle pattern:** Stable to slightly fluctuating pattern, as evident from features Transmission Time [0.248 to 0.402], showing minor fluctuation in delivery time.
- **End pattern:** Stable to slightly fluctuating pattern, based on features Transmission Time [0.320 to 0.256], suggesting slight fluctuations continue.
- **Overall trend:** Stable, indicating the presence of high-throughput conditions.

Viewer’s video buffer:

- **Initial pattern:** Stable pattern, as observed from the features Client Buffer [15.000 consistently], indicating full buffer.
- **Middle pattern:** Stable to slightly decreasing pattern, as evident from features Client Buffer [15.000 to 14.951], suggesting minor depletion.
- **End pattern:** Slightly increasing to stable pattern, based on features Client Buffer [14.951 to 15.000], showing recovery to full capacity.
- **Overall trend:** Stable, indicating the presence of robust buffer capacity.

Viewer’s Quality of Experience (environment reward or stalling behavior):

- **Initial pattern:** Increasing pattern, as observed from the features Quality of Experience [3.033 to 3.223], indicating improving experience.
- **Middle pattern:** Decreasing to increasing pattern, as evident from features Quality of Experience [3.223 to 2.974], showing temporary dip.
- **End pattern:** Increasing to stable pattern, based on features Quality of Experience [2.974 to 3.064], indicating recovery to stability.
- **Overall trend:** Stable, indicating the presence of acceptable conditions.

Upcoming video sizes:

- **Initial pattern:** Increasing pattern, as observed from the features Mean Upcoming Video Sizes [0.272 to 0.874], suggesting increasing sizes.
- **Middle pattern:** Increasing to stable pattern, as evident from features Upcoming Video Sizes [0.874], showing stabilization at higher values.
- **End pattern:** Consistent pattern, based on features Mean Upcoming Video Sizes [0.874], indicating no further change.
- **Overall trend:** Increasing, indicating the presence of higher quality content.

Upcoming video qualities:

- **Initial pattern:** Decreasing pattern, as observed from the features Mean Upcoming Video Qualities [15.949 to 10.676], indicating a reduction in quality.
- **Middle pattern:** Stabilizing pattern, as evident from features Mean Upcoming Video Qualities [10.676], reflecting adjustment.
- **End pattern:** Stable pattern, based on features Mean Upcoming Video Qualities [10.676], with no further decrease.
- **Overall trend:** Decreasing, indicating reduced quality expectations.

Summary:

- **Altogether, the patterns in the features indicate:** Stable network, stable client video player, and adjusting video stream patterns.
- **Key concept correlation:** Robust Buffer Capacity, Moderate Network Throughput, Smooth Switching Strategy, Stable, High Network Throughput, and Content Suitability.

Figure 16: The response for the example prompt given in Figure 15, following the structure and reasoning patterns of the prompt. The entire response (with the key concept correlation redacted) serves as our text description to measure similarity.