Approximating Pensieve with Classical Techniques

Thomas Schibler

Abstract

We present a framework for approximating the behavior of Pensieve, a reinforcement learning-based adaptive bitrate (ABR) streaming algorithm, using interpretable white-box models. By applying a closed-loop machine learning pipeline, we use a genetic algorithm to tune the parameters of classical models to mimic Pensieve's bitrate decisions on fixed network traces. This approach enables systematic evaluation of where simple models succeed or fail, offering insights into Pensieve's decision logic and guiding the design of more transparent, high-performing ABR strategies.

1 Introduction

Recent advances in machine learning have produced increasingly capable models, but also increasingly complex and opaque ones. While high performance is often achieved, it can come at the cost of interpretability, which is critical in many real-world and safety-critical applications. In such domains, we care not just about what decisions a model makes, but how and why it makes them, especially when facing unanticipated situations.

One example of this is adaptive bitrate (ABR) control in video streaming. ABR algorithms select video quality levels dynamically based on current network conditions, aiming to balance playback quality with stability. Traditional heuristics approach this by codifying simple human intuitions, such as choosing lower bitrates when buffer is low, which makes them easy to understand and anticipate. However, newer systems like Pensieve [4] (which uses reinforcement learning) outperform these classical approaches, but make decisions that are difficult to explain. We explore how well Pensieve can be approximated using interpretable, white-box models. Our goal is not just to see how well we can match performance, but to gain insight into what Pensieve has learned and where its performance gains come from. Does it rely on entirely novel principles, or does it refine existing heuristics in subtle ways?

In this project, we apply the closed-loop machine learning pipeline to iteratively refine a human-specified model until it best approximates Pensieve. Our system allows for white-box models (or ensembles) to be specified with tunable hyperparameters, which are optimized via the TOGA genetic algorithm to minimize mean absolute error against Pensieve's decisions on fixed network traces. A human evaluator can then assess the resulting model, identify gaps in performance or coverage, and return to the model design phase, enabling a tight human-in-the-loop development cycle.

At a high level, this work draws inspiration from Trustee [3], which provides model-agnostic surrogate explanations for black-box systems. Where Trustee is designed to generalize across domains, our work focuses specifically on domain-informed, human-specified heuristics, aiming to understand when and how such approaches can approximate sophisticated learned policies. In doing so, we hope to bring some of the benefits of explainability to high-performance systems like Pensieve, and to identify principled ways for improving classical models using insight from learned ones.

2 Background and Motivation

Adaptive bitrate (ABR) algorithms decide video quality during streaming to maximize user experience under changing network conditions. Classical approaches include buffer-based algorithms and model predictive control (MPC); see [2, 5, 6]. These are interpretable and simple but often underperform in complex environments.

Pensieve [4] improves performance by using reinforcement learning to learn bitrate decisions in simulation. However, its black-box nature makes it hard to understand or trust, especially in critical or unfamiliar scenarios.

Explainability tools like Trustee [3] approximate black-box models with interpretable surrogates. Trustee has the benefit of being problem-agnostic and makes it easy for users to see the main factors in decision making.

In contrast, this project applies the closed-loop machine learning pipeline to examine how well white-box ABR models built from human intuition specifically for ABR, can mimic Pensieve. In this way, we are not only trying to understand Penseive, but also see where our intuition falls short.

3 Design/Approach/Methodology

The core design of the system is centered around enabling a closed-loop machine learning pipeline for interpretable model development. At the center of the pipeline is a genetic algorithm called TOGA [1] that tunes candidate white-box models to replicate the bitrate decisions made by Pensieve over a fixed set of network traces. Each model is composed of a set of interpretable decision rules (e.g., threshold conditions, simple ensembles), and each model's hyperparameters and how to combine them are treated as tunable parameters within TOGA. The objective function guiding this evolution is the mean absolute error between the candidate model's output and Pensieve's decisions, averaged across all traces.

This structure enables a feedback loop: once a best-performing model is selected, a human-in-the-loop can analyze its structure to determine where it diverges from Pensieve's behavior. These gaps can highlight which decision factors (e.g., buffer occupancy, recent throughput, download variability) the white-box model might be missing. New features, rules, or model components can then be introduced into the system, and the pipeline is rerun, closing the loop and allowing for progressive refinement of interpretable policies through structured experimentation.

4 Implementation

Our implementation can be found at schibler/293Project. So far it is a minimal prototype; we demonstrate a working pipeline for a single white box model (the baseline buffer based (BB) approach used in the Penseive study), with a single hyperparameter (a linear weight on the BB model output). In this section, we describe the specific implementation challenges we faced, and end the section with an overview of possible future development.

Replicating Pensieve. To get to a prototype, we first needed to replicate Pensieve, which is implemented in Python and depends on now outdated packages (in particular, Tensorflow 1.X). Unfortunately, these predate the Apple M1 chip (our replication target) and are not supported. To get around this, we built a Docker image which (more closely) matches Pensieve's original target, with the exception of upgrading from Python 2 to Python 3. As a consequence, we could only run Pensieve using CPU, so we limited our scope to working with the pretrained version of Pensieve included in the original repository, in order to avoid an expensive training step. Furthermore, we only worked with Pensieve in simulation.

Secondly, we needed to recover the input data used to simulate Pensieve. This includes 1) a (chunked) sample video file to download, and 2) "cooked" network traces indicating the available bandwidth for use by the simulator. We reused the sample video file included in Pensieve, but could not locate the cooked trace artifacts used in the original Pensieve study. To replicate this data, we worked used a subset of the raw data used in the study, namely, the Belgium 4G/LTE bandwidth logs. We implemented a script to "cook" the raw data to produce a static network trace for use by the simulator, which specifies a timeseries of the available throughput in Mbps at regular time intervals. We then ran the Pensieve simulator with our cooked traces to produce bit rate predictions at each time step, which one to one correspond with the trace throughputs. Of course, the exact trace/ABR prediction pair is modular, and it is easy to run on other data.

Specifying white box models. To effectively compare the performance of a white box model (or ensemble), we needed to run said model in an identical simulation loop as Pensieve. We again relied on an implementation of the original Pensieve work of a buffer based protocol. This protocol dynamically reduces bit rate as the available buffer size decreases, in order to avoid rebuffering, and increases when the buffer is full to maximize quality. We modified this protocol with a "weight" parameter, that linearly scales the bit rate by a fixed, tunable constant. In this way, we introduce a hyperparameter to control how "aggressive" the scheme is with respect to achieving quality at risk of re-buffering.

Tuning with Toga. Finally, we used the Tuning Optimizing Genetic Algorithm (TOGA) to run hyperparameter tuning on the white box, buffer based protocol. TOGA uses a client/server protocol, where the server instantiates sets of hyperparameters, for which any number of connected clients may work to evaluate. The server combines high performing hyperparameters via user specified crossover/mutation rules, keeping track of top performers, while (optionally) maintaining diversity across select hyperparameters. This means that we needed to define a fitness metric to fix what "high performing" means in the context of ABR prediction, and give TOGA a wrapper to evaluate the fitness of an arbitrary hyperparemter setting. We implemented this via a simple wrapper around the Pensieve simulation loop, defining fitness to be the mean absolute error between the raw bitrate predictions in Mbps, however, this choice is modular.

5 Evaluation

For our prototype, we show how the mean absolute error between the buffer-based protocol's ABR choices and Pensieve's choices change as we scale the formers prediction. It is worth pointing out that this scaling is done *in the simulation loop*, which means is not simply a linear fit after the fact. Indeed, the the choice of bit rate has a noticable effect on the buffer size at the next simulation step, which in turn impacts the protocols next decision.

Our sample results are shown in figure 1. In this experiment, TOGA was instructed to keep track of top performers in each length 0.5 interval of the weight hyperparameter in the range [0-5]. The plot shows the top performance (minimum error) within each bucket.

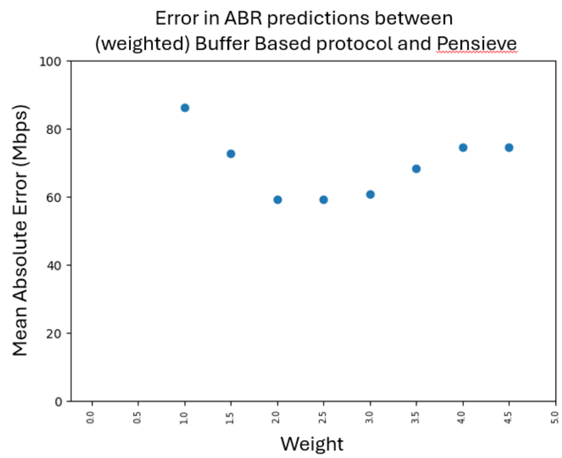

Figure 1: *Mean absolute error between top performing buffer based protocols and Pensieve shows that Pensieve is more aggressive in maximizing video quality. In particular, error is minimized when the buffer based approach is scaled 2-3x in the simulation loop, with performance decreasing to either side.*

6 Conclusion

We implemented a prototype framework for iteratively approximating Pensieve with classical techniques. As a preliminary result, we showed that Pensieve is more aggressive in maximizing quality when compared to the baseline buffer-based approach used by the Pensieve authors. Much future work remains: it would be interesting to see what additional hyperparameters and white-box models are needed to better match Pensieve's performance. Furthermore, this methodology can be applied to other predictive tasks where a fixed black box model can be used to generate labeled data.

References

[1] (Cited on page 1.)

[2] HUANG, T.-Y., JOHARI, R., MCKEOWN, N., TRUNNELL, M., AND WATSON, M. A buffer-based approach to rate adaptation: evidence from a large video streaming service. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (New York, NY, USA, 2014), SIGCOMM '14, Association for Computing Machinery, p. 187–198. (Cited on page 1.)

[3] JACOBS, A. S., BELTIUKOV, R., WILLINGER, W., FERREIRA, R. A., GUPTA, A., AND GRANVILLE, L. Z. Ai/ml for network security: The emperor has no clothes. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2022), CCS '22, Association for Computing Machinery, p. 1537–1551. (Cited on page 1.)

[4] MAO, H., NETRAVALI, R., AND ALIZADEH, M. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (New York, NY, USA, 2017), SIGCOMM '17, Association for Computing Machinery, p. 197–210. (Cited on page 1.)

[5] SPITERI, K., URGAONKAR, R., AND SITARAMAN, R. K. Bola: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking 28*, 4 (2020), 1698–1711. (Cited on page 1.)

[6] YIN, X., JINDAL, A., SEKAR, V., AND SINOPOLI, B. A control-theoretic approach for dynamic adaptive video streaming over http. *SIGCOMM Comput. Commun. Rev. 45*, 4 (August 2015), 325–338. (Cited on page 1.)