Neural ABR Deployment Decision Sheet

Pensieve · Comyco · Kairos · ABRL · Fugu · SODA — when neural ABR ships and when it does not

Four deployment shapes where neural ABR wins

Shape	When it wins
Large trace corpus	**Operator has hundreds of thousands of real-user session traces (Netflix, YouTube, Disney+, Meta, Amazon, large MVPDs). Smaller operators rarely meet the data bar without a vendor.**
Characterisable network	**Wired residential broadband, fixed wireless, characterised mobile networks. Heterogeneous mobility (satellite, 4G↔5G handoffs, public Wi-Fi at venues) generalises poorly out of the box.**
Switch suppression is tracked	**Apps where the product team measures switches per minute as a brand quality benefit disproportionately — Comyco, Kairos, SODA all suppress switches more aggressively than simple rules.**
Team to maintain the model	**Production deployments need a retraining pipeline, A/B infrastructure that detects single-digit-percent shifts, observability tooling, and a deterministic fallback. Without those, lifecycle cost dwarfs QoE gain.**

Four production failure modes

Failure mode	Symptom & fix
Trace distribution mismatch	**Policy trained on one network distribution running on another. Pensieve underperformed Fugu in the Puffer trial for exactly this reason. Fix: retrain in situ on the deployment's own traces.**
Training cost & operational drag	**RL training is brittle; reward shaping is an art; fresh corpora can require thousands of GPU-hours to retrain. Imitation learning (Comyco, Fugu) is dramatically cheaper but still needs an expert and a maintained pipeline.**
Opacity in production debugging	**A neural policy is a black box. Build observability tooling — saliency maps, decision logging, counterfactual replays — before deployment, not after the first customer escalation.**
Reward-function misalignment	**A policy maximises the reward you train it on. Engagement-shaped rewards (switches, TTFF, time-to-1080p) beat QoE-shaped rewards in production. SODA's case study makes this argument explicitly.**

Five tuning levers that actually move QoE

Lever	How to set it
Training method	**RL when no expert is available (Pensieve, ABRL). Imitation learning when an offline expert exists (Comyco, Fugu). Imitation is the 2026 default.**
Where the model sits	**Pure policy (Pensieve, ABRL, Comyco) when team can absorb opacity. Learned predictor inside an MPC controller (Kairos, Fugu) for production debugging. Hybrid framing is the 2026 default.**
Reward shaping	**Academic default: quality − λs·switches − λr·rebuffer. Production: add TTFF penalty, time-to-1080p penalty, viewing-duration term. Retrain when weights change.**
Retraining cadence	**Monthly for stable wired networks; weekly for fast-evolving mobile; continuous lifelong learning (L-Comyco) where the pipeline supports it.**
Fallback strategy	**Mandatory deterministic fallback to BOLA or MPC when the policy returns a pathological action. Log the incident, A/B detect the drift, retrain or roll back.**

Three production deployments documented at scale

Deployment	Architecture	Production result
Facebook ABRL (2020)	RL policy. 30M+ sessions/week.	**+1.6% avg bitrate · −0.4% stalls. arXiv 2008.12858.**
Stanford Puffer / Fugu (NSDI 2020)	MPC + in-situ learned predictor. 8,131 hours · 3,719 users.	**0.13% stall ratio · beats Pensieve in the wild. nsdi20-paper-yan.**
Amazon Prime Video SODA (SIGCOMM 2024)	Smoothness-optimised DP controller with QoE guarantees.	**−88.8% switches · +5.91% view duration on production fleet.**