

# BRAIN: Bayesian Reasoning via Active Inference for Explainable Mobile Networks

Osman Tugay Basaran<sup>\*†‡</sup>, Martin Maier<sup>†</sup>, and Falko Dressler<sup>\*</sup>

<sup>\*</sup>School of Electrical Engineering and Computer Science, TU Berlin, Germany

<sup>†</sup>Optical Zeitgeist Laboratory, INRS, Montréal, QC, Canada

<sup>‡</sup>Fraunhofer Heinrich-Hertz-Institut, Berlin, Germany

Email: {basaran, dressler}@ccs-labs.org, {martin.maier}@inrs.ca

**Abstract**—Emerging sixth-generation (6G) solutions demand learning agents that are not only autonomous and efficient, but also robust in dynamic environments and transparent in decision making. However, conventional deep reinforcement learning (DRL) approaches for mobile networks lack sufficient explainability and suffer catastrophic forgetting under dynamic conditions. In this paper, we propose a Bayesian reasoning-based explainable deep active inference (BRAIN) model, the embodied AI-inspired approach applied to mobile networks. BRAIN employs a deep generative model and minimizes variational free energy to unify perception (Bayesian state estimation) and action (resource allocation) in a single framework. Unlike DRL baselines, our agent inherently eliminates catastrophic forgetting through continuous belief updates without retraining and provides built-in explainability by exposing posterior beliefs and free energy components at runtime. Deployed as an eXtended application (xApp) on an Open, GPU-accelerated, AI-RAN testbed; BRAIN demonstrates (i) causal reasoning for slice-specific quality of service (QoS) adherence (throughput/latency/reliability), (ii) 28.3% higher robustness to traffic shifts versus DRL baselines, and (iii) real-time interpretability of resource-allocation decisions via operator-accessible belief states.

**Index Terms**—6G, AI-RAN, mobile networks, active inference, explainability

## I. INTRODUCTION

Future sixth-generation (6G) mobile networks are envisioned to be artificial intelligence (AI)-native, tightly integrating AI for real-time optimization and control [1]. Current AI-driven solutions largely rely on established neural-network models (recurrent neural networks [2] or autoencoders [3]) to tackle wireless tasks. While these deep learning-based approaches have shown initial promise, they inherit fundamental limitations when confronted with the full complexity of real-world wireless environments. The wireless environment is inherently non-stationary considering that channel conditions, user mobility patterns, and traffic loads fluctuate continuously over time [4]. Models trained offline on static data often struggle to generalize across rapidly changing network states, leading to performance degradation unless they are frequently retrained or fine-tuned. In response to these challenges, researchers have explored reinforcement learning (RL) [5] and further deep reinforcement learning (DRL) [6] for wireless network control. DRL combines neural function approximation with closed-loop learning (see Fig. 1a), enabling agents to learn resource allocation and scheduling policies through direct interaction with the network environment.

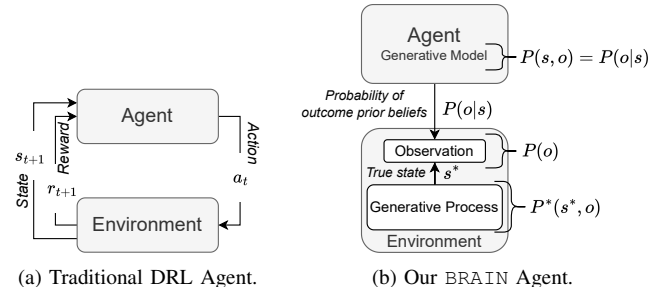


Fig. 1: Architecture of conventional DRL and proposed explainable deep active inference agents.

Indeed, DRL-driven agents have demonstrated efficacy in problems such as radio resource management, user scheduling, and network slicing. However, a critical shortcoming of standard DRL solutions is their lack of explainability. Trained DRL policies operate as black boxes that lack interpretability, making it difficult for network operators to trust and diagnose their behavior. Beyond transparency, conventional DRL approaches are notoriously vulnerable to catastrophic forgetting when exposed to new network scenarios. A DRL agent can rapidly overwrite and forget previously learned behaviors. Standard DRL lacks a mechanism for graceful adaptation to distribution shifts; it must relearn from scratch whenever network conditions deviate substantially from those seen during training. These limitations motivate the research for fundamentally more robust and interpretable AI foundations. Instead of viewing network control purely as a data-driven function approximation problem, there is a growing consensus that next-generation AI agents need cognitive capabilities such as reasoning and knowledge integration [7]. In particular, incorporating causal reasoning principles is seen as a promising path to enhance robustness and interpretability. Causal reasoning endows an AI agent with an understanding of cause-effect relationships, moving beyond pattern recognition [8].

In this context, we propose to leverage the Bayesian reasoning via active inference as a principled alternative to DRL for explainable mobile networks. Active inference, originally developed in the field of cognitive neuroscience, offers a unifying theory of perception, action, and learning based on variational Bayesian inference [9]. To the best of our knowledge, this paper is the first to introduce the active inference

framework in the context of intelligent resource management for next-generation mobile networks. We posit that our BRAIN agent model (see Fig. 1b) can provide a novel alternative to conventional DRL-based network control, one that addresses the twin challenges of robustness and explainability in future self-organizing mobile networks.

The core outcomes of our study are summarized as new contributions (“C”) and new findings (“F”) as follows:

- C1.** We present the first explainable active inference learning approach applied in mobile networks.
- C2.** We demonstrate in a GPU-enabled testbed causal reasoning and slice-specific QoS adherence in real time.
- F1.** Belief updates naturally handle distribution shifts and eliminate catastrophic forgetting.
- F2.** Posterior beliefs and free-energy components are exposed at runtime, offering operators clear, causal insight into each resource-allocation decision.

## II. EXPLAINABLE DEEP ACTIVE INFERENCE DESIGN

### A. Generative Model Design

We define a set of hidden state variables  $s_t$  that characterize the real-time status of the network and its slices at time  $t$ .  $s_t$  include latent features such as per-slice traffic load, channel quality indicators, or queue lengths for each network slice. The agent receives observations  $o_t$  (e.g., measured throughput, latency, or slice performance metrics) that are probabilistically generated from the hidden state. The agent can also perform actions  $a_t$  (or controls) corresponding to slice resource allocation decisions. Without loss of generality, we refer to scenarios such as distributing a limited pool of physical resource block (PRB) among slices or selecting scheduling policies for each slice, as typically encountered in O-RAN systems. In Algorithm 1, BRAIN’s generative model is expressed as a joint distribution over states, observations, and actions across time,  $P(s_{0:T}, o_{0:T}, a_{0:T})$ , which factorizes into dynamical and observational components:

$$P(s_{0:T}, o_{0:T}, a_{0:T}) = P(s_0) \prod_{t=0}^{T-1} P(o_t | s_t), P(s_{t+1} | s_t, a_t), P(a_t). \quad (1)$$

### B. Variational Inference and Policy Selection

*Perception as variational inference:* Upon receiving a new observation  $o_t$  (e.g., current slice throughput levels), the agent must infer the latent state  $s_t$  (e.g., actual demand causing those throughput levels). BRAIN performs this by minimizing the variational free energy  $F_t$ , which serves as a proxy for Bayesian inference. We define the free energy at time  $t$  as:

$$F_t(Q(s_t)) = \mathbb{E}Q(s_t)[- \ln P(o_t | s_t) - \ln P(s_t | s_{t-1}, a_{t-1})] + \mathbb{H}[Q(s_t)], \quad (2)$$

where  $\mathbb{H}[Q]$  is the entropy of the approximate posterior  $Q$ . Intuitively, minimizing  $F_t$  encourages the posterior  $Q(s_t)$  to explain the observation well (high likelihood  $P(o_t | s_t)$ ) while staying close to the prior prediction  $P(s_t | s_{t-1}, a_{t-1})$

---

### Algorithm 1: Proposed BRAIN Agent

---

**Input:** Generative model  $P(s_{t+1}, o_{t+1} | s_t, a_t)$ ;  
 Preference distribution  $P_{\text{pref}}(o)$ ; Action set  $\mathcal{A}$ ;  
 Time horizon  $T$ ; Prior belief  $Q(s_0)$ .

**Output:** Selected action sequence  $\{a_0, a_1, \dots, a_{T-1}\}$ ;  
 Logged beliefs and free-energy terms for explainability.

Initialize  $Q(s_0)$  (prior belief over states)

**for**  $t \leftarrow 0$  **to**  $T - 1$  **do**

$o_t \leftarrow$  observe new data from environment  
 // Receive observation  $o_t$  (e.g., current network slice metrics)

$Q(s_t) \leftarrow$  BayesianUpdate( $Q(s_{t-1}), o_t$ )

// Update posterior state belief given  $o_t$

// Belief update yields  $Q(s_t)$

**foreach**  $a_t \in \mathcal{A}$  **do**

Compute  $Q(o_{t+1} | a_t)$  using generative model  
 // Predict observation distribution if action  $a_t$  is taken

$\text{KL}_{\text{pref}} \leftarrow D_{\text{KL}}(Q(o_{t+1} | a_t) \parallel P_{\text{pref}}(o_{t+1}))$   
 // Divergence from preferred outcomes (extrinsic term)

$I_{\text{gain}} \leftarrow I(s_{t+1}; o_{t+1} | a_t)$   
 // Expected information gain (epistemic term)

$G(a_t) \leftarrow \text{KL}_{\text{pref}} - I_{\text{gain}}$   
 // Expected free energy  $G(a_t)$

$a_t^* \leftarrow \arg \min_{a_t \in \mathcal{A}} G(a_t)$   
 // Select action that minimizes expected free energy

Execute  $a_t^*$  on environment

// Apply chosen action (Adjust network slice resources)

Log  $Q(s_t)$ ,  $\{G(a) : a \in \mathcal{A}\}$ , and  $a_t^*$  for analysis

// Record beliefs and free-energy terms for explainability

---

(which is the generative model’s forecast from the previous time, ensuring temporal consistency). In practice, we assume a tractable form for  $Q(s_t)$  (i.e., Gaussian distribution or delta at a point estimate) and perform a gradient-based update (or closed-form update if linear-Gaussian) to find  $Q^*(s_t) \approx P(s_t | o_{\leq t}, a_{< t})$ . This posterior belief  $Q(s_t)$  encapsulates the agent’s current understanding of network conditions after seeing the data. Notably, because the generative model is explicit, this belief is fully transparent: each variable in  $s_t$  has a quantitative posterior that can be reported.

### C. Action Selection as Expected Free Energy Minimization

After updating its beliefs, the agent must decide on the next action  $a_t$  (e.g., how to reallocate resources among slices)

before the next observation arrives. Instead of using a learned Q-value or policy network as in RL, BRAIN evaluates prospective actions by their expected free energy  $G(\pi)$ , where  $\pi$  denotes a candidate policy (a sequence of future actions, or simply the single action  $a_t$  in a one-step horizon case). The agent chooses the policy that minimizes  $G(\pi)$ , reflecting the principle of active inference: actions are chosen to minimize expected surprise and fulfill preferences. We formulate  $G(\pi)$  (for simplicity, with a one-step horizon  $\pi = a_t$ ) as:

$$G(a_t) = \mathbb{E}Q(s_{t+1} | s_t, a_t) \left[ -\ln P(o_{t+1} | C) + D_{\text{KL}}(Q(s_{t+1} | o_{t+1}) | Q(s_{t+1})) \right], \quad (3)$$

where the expectation is taken with respect to the predicted next-state distribution under action  $a_t$ , and inside we consider two quantities for the next time step  $t + 1$ :

- ① Expected ‘‘Surprise’’ w.r.t Preferences:  $-\ln P(o_{t+1} | C)$  is the surprisal (negative log-probability) of a future observation  $o_{t+1}$  under the preference model. If an action is likely to produce outcomes aligned with  $C$  (e.g., all slices meet their targets), this term will be low; if the action would result in poor performance for some slice, yielding an observation far from the preferred range, this term will be high. Thus, the expected value  $\mathbb{E}[-\ln P(o_{t+1} | C)]$  serves as a risk or phenotypic cost for that action.
- ② Expected Information Gain (Epistemic Value): The second term  $D_{\text{KL}}[Q(s_{t+1} | o_{t+1}) | Q(s_{t+1})]$  is the KL divergence between the posterior and prior predicted state at  $t + 1$ , which quantifies how much we expect to learn about the hidden state by observing  $o_{t+1}$ . Equivalently, it can be expressed as the mutual information  $I[s_{t+1}; o_{t+1}]$  between state and observation under that action. An action that is expected to resolve uncertainty (e.g., probing an uncertain slice’s condition by allocating it resources to see if throughput improves) will have a high information gain, reducing  $G$ .

#### D. Introspective Explainability of Decisions

At each time  $t$ , the BRAIN agent maintains a posterior belief distribution over the latent slice states  $s_t$  (e.g., each slice’s current demand level or reliability). We denote this belief as:

$$Q(s_t) = P(s_t | o_{1:t}, a_{1:t-1}), \quad (4)$$

with the probability of each hidden state given all past observations  $o_{1:t}$  and actions  $a_{1:t-1}$ . In practice,  $Q(s_t)$  is computed via the agent’s variational Bayes update after receiving observation  $o_t$ . For example, if slice demand can be high or low,  $Q(s_t)$  might be a probability  $P(s_t = \text{high})$  or  $P(s_t = \text{low})$  that is updated as new traffic measurements come in. These posterior beliefs are introspective variables because they represent the agent’s internal knowledge about the network (and can be exposed for explainability). The distribution  $Q(s_t)$  is normalized and reflects the agent’s confidence in different slice conditions at time  $t$ . The agent encodes prior preferences about desirable outcomes for the network slices. Let  $P_{\text{pref}}(o)$  denote the preferred distribution over

observations (or performance metrics).  $P_{\text{pref}}(o)$  could assign high probability to outcomes where each slice’s throughput or latency meets its target. At each decision step, the agent predicts the distribution of next observations  $Q(o_{t+1} | a_t)$  for a candidate action  $a_t$  (by marginalizing over its state beliefs). We define a preference alignment cost as the Kullback–Leibler (KL) divergence between the predicted outcome distribution and the preferred distribution. Formally, for action  $a_t$ :

$$D_{\text{KL}}\left(Q(o_{t+1} | a_t) \middle| P_{\text{pref}}(o_{t+1})\right) = \sum_{t=0}^{\infty} Q(o_{t+1} | a_t) \ln \frac{Q(o_{t+1} | a_t)}{P_{\text{pref}}(o_{t+1})}. \quad (5)$$

This term measures how well the agent’s expected observations align with its preferences (a form of phenotypic risk). Minimizing this term drives the agent to choose actions that fulfill slice requirements (extrinsic reward-seeking behavior). In parallel, the agent quantifies the epistemic value of actions, i.e., the expected information gain about the latent slice states. This captures the action’s exploratory benefit: how much taking action  $a_t$  will reduce uncertainty in  $s_{t+1}$ . Mathematically, we can define the epistemic value as the mutual information between future states and observations, conditioned on  $a_t$ . One convenient form is the expected reduction in entropy of the state-belief after observing  $o_{t+1}$ :

$$I(s_{t+1}, o_{t+1} | a_t) = H[Q(s_{t+1} | a_t)] - \mathbb{E}Q(o_{t+1} | a_t) \left[ H[Q(s_{t+1} | a_t, o_{t+1})] \right], \quad (6)$$

where  $H[Q(s)] = -\sum_s Q(s) \ln Q(s)$  is the entropy. Intuitively,  $H[Q(s_{t+1} | a_t)]$  is the agent’s prior uncertainty about the next state before taking action  $a_t$ , and the second term is the expected posterior uncertainty after seeing the new observation. Their difference is the expected information gain. If action  $a_t$  is purely informational (i.e., a probing action that reveals a slice’s condition), this quantity will be high, meaning the agent anticipates learning a lot (large epistemic value). If an action is not informative, i.e., it does not affect observations, the epistemic value is low. In many active inference formulations, this corresponds to reducing *perceptual ambiguity*.

### III. EXPERIMENT DESIGN

#### A. Intelligent Orchestration on AI-RAN Testbed

We deploy a private 5G testbed (see Figure 2) featuring a GPU-accelerated O-RAN architecture built on the NVIDIA Aerial Research Cloud (ARC) platform [10, 11] and Aerial SDK [12]. In our setup, the gNB’s protocol stack is split into an O-DU Low (Layer-1 PHY) running on an NVIDIA GPU and an O-DU High/CU (higher layer protocols) running on x86 CPUs with OpenAirInterface (OAI) [13]. The testbed is equipped with both commercial and softwarized UEs to generate multi-slice traffic. In particular, we use a COTS 5G UE (Sierra Wireless EM9191 modem module) and an OAI-based soft UE (nrUE) as two end devices. We consider a multi-slice RAN control scenario with GPU-Accelerated gNB (the testbed

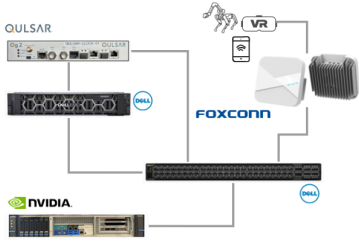


Fig. 2: Overview of GPU-Accelerated AI-RAN Testbed.

described above, with one O-RU) serving UEs and orchestrating 3 slice types: enhanced Mobile Broadband (eMBB), Ultra-Reliable Low-Latency Communications (URLLC), and Massive Machine-Type Communications (MTC). Each slice is configured with its own QoS requirements and traffic profile (e.g., high throughput for eMBB, low-latency for URLLC, low-rate internet of things (IoT) traffic for MTC).

To enable closed-loop control, we define the RAN state (observation) in terms of key O-RAN key performance measurements (KPMs). In particular, the state includes slice-level downlink throughput (DL data rate), buffer occupancy (downlink queue length), and the count of transmitted PHY transport blocks (TBs) in the downlink. These metrics are continuously collected from the gNB via the O-RAN E2 interface using the standardized KPM service model. A dedicated KPM monitoring eXtended application (xApp) on the Near-RT RIC [14] subscribes to these telemetry streams, functioning as an observer that writes the latest KPM values to the RIC’s data store (shared data layer). The choice of state features follows prior work on RAN slicing control [15]: throughput, queue occupancy, and TB count have been shown to capture the essential traffic demand and quality conditions for each slice while remaining low-dimensional enough for efficient learning. Action space for slice control consists of adjusting the PRB allocation and scheduling policy for each slice on the gNB. Specifically, our controller can dynamically set the fraction of PRBs (resource blocks) allocated to each slice (effectively, a slicing policy that partitions the 100 MHz bandwidth among eMBB, URLLC, and MTC) and select the packet scheduling algorithm employed per slice. In our implementation, we consider a discrete set of scheduling policies such as proportional fair (PF), round robin (RR), and weighted fair queuing (WFQ), which can be assigned independently to each slice. BRAIN xApp runs on the Near-RT RIC (with control periodicity on the order of tens of milliseconds) and observes the aforementioned KPM state. At each control interval, it issues an action: an updated allocation of PRBs to slices and/or a scheduler selection for each slice.

### B. Benchmark Models

**Deep Q-Network (DQN) Agent:** A value-based RL agent that learns a Q-value for each possible action from each state [16]. DQN uses an  $\epsilon$ -greedy exploration policy (with  $\epsilon$  annealed from 1 to 0.1 over training) and a replay buffer to store past experiences. We use a deep neural network (5 fully-connected layers, 30 neurons each,  $\tanh$  activations) to

approximate the Q-function, following common practice. Key hyperparameters include ① Learning rate of  $10^{-3}$  and ② Discount factor of  $\gamma = 0.99$ . ③ Replay memory can hold up to  $10^5$  state-transition samples, and the DQN is trained with ④ Mini-batches (size 64) drawn from this buffer at each learning step. A target network is used for stability, updated every 100 training iterations.

**Actor-Critic (AC) Agent:** A policy-based RL agent that has an actor-critic architecture [17]. The actor network directly outputs the slicing action probabilities (or a deterministic action in a continuous parameterization), while the critic network estimates the value (expected return) of the current state (or state-action pair). We implement a synchronous advantage AC algorithm: the actor updates its policy in the direction suggested by the critic’s advantage estimates, and the critic learns to predict returns to reduce the temporal-difference error. Both actor and critic are ① 5-Layer fully-connected neural networks (sharing the first few layers in our implementation) with ② 30 neurons per layer. We use ③ Learning rate of  $10^{-3}$  for both networks and ④  $\gamma = 0.99$  for reward discounting.

**Policy Gradient (PG) Agent:** A baseline agent that uses vanilla PG reinforcement learning (specifically, REINFORCE [18]). PG agent has a single policy network (again ① 5-Layer MLP with ② 30 neurons each) that directly outputs the probability distribution over actions (or continuous action parameters). It updates the policy by computing the gradient of the expected reward and adjusting weights in the direction that increases the probability of successful actions. To reduce variance, we utilize a baseline subtraction (average reward or a simple critic) when computing PGs, but unlike the full AC, we do not train a separate value network and the focus is on a simpler policy optimization approach. We use a slightly smaller ③ Learning rate ( $5 \times 10^{-4}$ ) for the policy network to ensure stable convergence, given the higher variance of PG updates, and ④  $\gamma = 0.99$ .

## IV. EVALUATION

**Explainability Analysis.** We model each slice’s demand as a hidden state (Low/Medium/High) and visualize the agent’s posterior belief over time as heatmaps in Fig. 3. In the eMBB Slice (Fig. 3a), the agent quickly concentrates its belief on the High demand state (bright band in the top row) once high traffic is observed. Around  $t \approx 40$ , the underlying demand drops and the belief becomes less concentrated, spreading across states before refocusing on Medium. After the arrow-labeled Check actions, the belief sharpens again and returns to High by  $t \approx 70$ . This matches the intuition that the agent becomes more certain when informative observations arrive (a single dominant bright band) and remains more uncertain otherwise (belief distributed across multiple states). In Fig. 3b, the posterior initially alternates mainly between High and Low, reflecting ambiguity in early observations. At  $t \approx 30$ , a demand shift drives rapid belief concentration toward Low. Following the epistemic Check at  $t \approx 60$ , the belief becomes more peaked and consolidates strongly on Medium

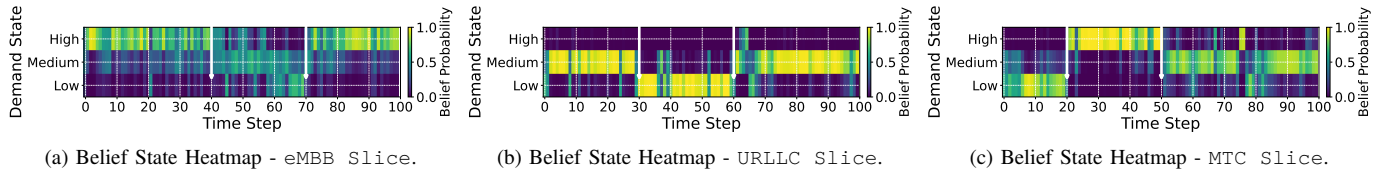


Fig. 3: Agent’s posterior belief trajectory over hidden traffic demand levels (Low, Medium, High) for each network slice across episodes. Time on the x-axis indexes discrete decision epochs (e.g., observation-update intervals  $\times 10^3$ ), and the y-axis enumerates the three demand states (Low at bottom to High at top). Color encodes the belief probability  $Q(s_t)$  (*brighter/yellow = higher; darker/purple = lower*). White arrows mark the agent’s explicit information-gathering Check actions.

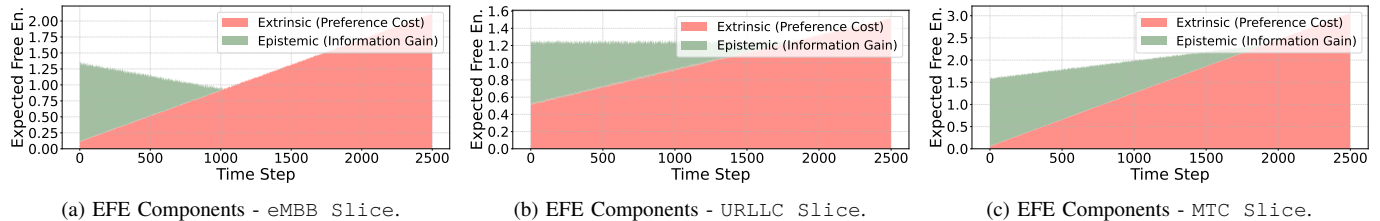


Fig. 4: Interpretation of the expected free energy (EFE) ( $G(\pi)$ ) decomposition. At each time step, the chosen action’s epistemic (soft green) and extrinsic (soft red) terms presented. Epistemic value dominates early on (favoring exploration) and then gives way to extrinsic value (favoring QoS/exploitation).

demand. In Fig. 3c, the agent begins with low uncertainty, confidently focusing on Low. After Check at  $t \approx 20$ , the belief shifts decisively toward High. Around  $t \approx 50$ , changing conditions increase uncertainty (belief spreads across states) before gradually concentrating toward the Medium state by  $t \approx 70$ . In Fig. 4a for the eMBB Slice, we observe that the epistemic value dominates in the early phase, where the green area is most prominent. This indicates that the agent is initially exploring uncertain aspects of eMBB traffic demands, likely performing observation-driven or probing actions to refine its internal beliefs about bandwidth requirements. Over time, the epistemic term steadily declines, while the extrinsic cost increases. This transition reflects that the agent has gained enough confidence in its beliefs and begins to shift toward exploitative behavior, focusing on aligning slice resource allocations with performance preferences. In Fig. 4b for the URLLC Slice, a slightly different pattern emerges. The epistemic and extrinsic components are more balanced during the early stages, implying that the agent simultaneously explores and regulates URLLC’s latency-critical requirements. This behavior reflects the tight QoS constraints of URLLC, which necessitate that even early decisions consider extrinsic risks. In Fig. 4c for MTC Slice, we see the strongest and longest-lasting epistemic engagement. The green region dominates the first half of the plot, suggesting that the agent initially dedicates extensive exploration effort to understand MTC’s demand dynamics, which are likely bursty and sparse in nature. After  $t = 2000$ , a sharp increase in extrinsic value occurs as the agent begins enforcing goal-directed behavior.

**Comparison with Benchmarks.** In Fig. 5a, we observe that BRAIN exhibits significantly faster convergence and sustained high performance relative to the baselines. The agent starts

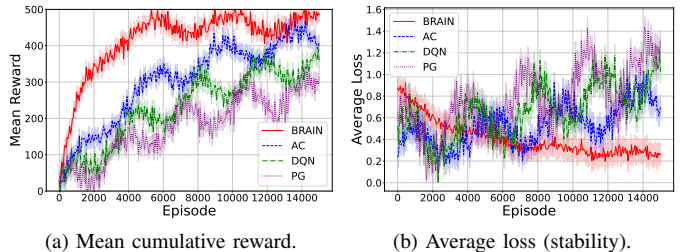


Fig. 5: Benchmark results with baseline models.

from zero reward like all others, but it rapidly improves and achieves near-peak performance within 3000 episodes. In contrast, the DRL models improve more slowly and plateau at lower reward levels. This behavior reflects BRAIN’s inherent ability to balance exploration and exploitation using its internal belief updates, allowing it to gather information about traffic dynamics more efficiently. Notably, AC reaches its saturation point around episode 6000, while DQN and PG continue fluctuating with increasing volatility. These fluctuations reveal the instability often seen in DRL under non-stationary conditions like demand variations across slices. Turning to Fig. 5b, while BRAIN maintains a smooth and steadily declining loss curve (converging to a low plateau) each DRL baseline displays erratic and increasing loss trends over time. For instance, both DQN and PG suffer from rising loss values after around episode 7000, indicating *instability and potential catastrophic forgetting*. BRAIN’s lower and flatter loss profile is directly attributed to its Bayesian update mechanism, which preserves posterior beliefs rather than discarding them. This allows the agent to be aware of prior experiences even as it encounters new dynamics. Quantitatively, BRAIN achieves a 28.3%

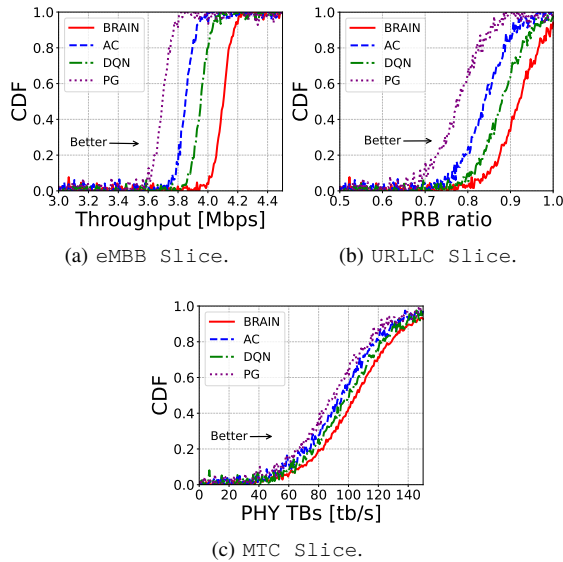


Fig. 6: Comparative slicing performance of the proposed BRAIN agent versus DRL baselines (AC, DQN, PG) across three different network slices.

higher mean reward than the best DRL baseline (AC). Besides, variance remains tightly bounded, with less than  $\pm 0.05$  spread throughout training. Compared to Q-Learning and Policy Gradient, BRAIN reduces loss magnitude by 40%, confirming better stability under non-stationary traffic conditions.

**Slicing Performance.** In Fig. 6a, results clearly show that BRAIN consistently delivers higher and more stable throughput, with its curve tightly concentrated around 3.9–4.0 Mbps. The upper percentiles (80–90<sup>th</sup> percentiles) further accentuate BRAIN’s effectiveness, surpassing the baselines substantially and delivering throughput consistently above 4.1 Mbps. This more compact distribution for BRAIN demonstrates its ability to provide consistent quality of service (QoS) for high-bandwidth applications by rapidly adjusting PRB allocation based on updated belief states, avoiding the oscillatory patterns common in DRL policies. Fig. 6b presents the PRB allocation ratio for URLLC Slice, which directly reflects the ability to maintain low-latency and highly reliable communication under dynamic multi-slice load. BRAIN achieves a median PRB ratio above 0.88, with 95% of its samples surpassing 0.85. This consistent prioritization ensures that URLLC retains sufficient bandwidth even during congestion, preventing latency spikes. In Fig. 6c, we analyzed the system’s responsiveness to sporadic IoT traffic bursts. While all agents ultimately achieve similar maximum rates (140 tb/s), BRAIN reaching the 50% completion mark at around 60 tb/s, compared to 70–80 tb/s for the DRL baselines. This earlier ramp-up suggests that BRAIN reacts faster to sudden MTC Slice load increases, allowing it to meet reliability requirements.

## V. CONCLUSION

We introduced BRAIN, an explainable deep active inference agent for next-generation RAN slicing, and demonstrated its

advantages over baseline models. By minimizing expected free energy, our agent tightly couples perception and action, yielding decisions that are not only robust to changing network conditions but also transparently explainable. BRAIN’s deep generative model avoids catastrophic forgetting with continual (lifelong) learning capabilities and produces explicit posterior beliefs and decision metrics for operator scrutiny.

## ACKNOWLEDGMENT

The research was supported in part by the BMFTR xG-RIC project under grant number 16KIS2434.

## REFERENCES

- [1] W. Saad, O. Hashash, C. K. Thomas, C. Chaccour, M. Debbah, N. Mandayam, and Z. Han, “Artificial General Intelligence (AGI)-Native Wireless Systems: A Journey Beyond 6G,” *Proceedings of the IEEE*, vol. 113, no. 9, pp. 849–887, Sep. 2025.
- [2] O. T. Basaran, H. Zafar, M. Kasparick, F. Dressler, and S. Stańczak, “Next-Gen AI-on-RAN: AI-native, Interoperable, and GPU-Accelerated Testbed Towards 6G Open-RAN,” in *IEEE International Conference on Communications (ICC 2025)*, Montréal, Canada: IEEE, Jun. 2025, pp. 5362–5367.
- [3] O. T. Basaran and F. Dressler, “XAIomaly: Explainable and Interpretable Deep Contractive Autoencoder for O-RAN Traffic Anomaly Detection,” *Computer Networks*, vol. 261, p. 111145, Apr. 2025.
- [4] X. Cheng, Z. Huang, and L. Bai, “Channel Nonstationarity and Consistency for Beyond 5G and 6G: A Survey,” *IEEE Communications Surveys & Tutorials*, vol. 24, no. 3, pp. 1634–1669, 2022.
- [5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998, p. 322.
- [6] F.-L. Vincent, H. Peter, I. Riashat, G. B. Marc, and P. Joelle, “An Introduction to Deep Reinforcement Learning,” *Foundations and Trends® in Machine Learning*, vol. 11, no. 3–4, pp. 219–354, Dec. 2018.
- [7] C. K. Thomas, C. Chaccour, W. Saad, M. Debbah, and C. S. Hong, “Causal Reasoning: Charting a Revolutionary Course for Next-Generation AI-Native Wireless Networks,” *IEEE Vehicular Technology Magazine*, vol. 19, no. 1, pp. 16–31, Mar. 2024.
- [8] L. Bariah and M. Debbah, “AI Embodiment Through 6G: Shaping the Future of AGI,” *IEEE Wireless Communications*, vol. 31, no. 5, pp. 174–181, Oct. 2024.
- [9] K. J. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, J. O’Doherty, and G. Pezzulo, “Active inference and learning,” *Neuroscience & Biobehavioral Reviews*, vol. 68, pp. 862–879, Sep. 2016.
- [10] A. Kelkar and C. Dick, “Aerial: A GPU Hyperconverged Platform for 5G,” in *ACM SIGCOMM 2021, Demo Session*, Virtual Conference: ACM, Aug. 2021, pp. 79–81.
- [11] A. Kelkar and C. Dick, “NVIDIA Aerial GPU Hosted AI-on-5G,” in *4th IEEE 5G World Forum (5GWF 2021)*, Montréal, Canada: IEEE, Oct. 2021, pp. 64–69.
- [12] NVIDIA. “NVIDIA Aerial SDK.” [Online]. Available: <https://developer.nvidia.com/aerial-sdk>. (accessed: 15.01.2026).
- [13] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T.-T. Nguyen, “OpenAirInterface: Democratizing innovation in the 5G Era,” *Elsevier Computer Networks*, vol. 176, p. 107284, Jul. 2020.
- [14] O-RAN SC. “OSC Near Realtime RIC.” [Online]. Available: <https://wiki.o-ran-sc.org/display/RICP/2022-05-24+Release+E>. (accessed: 15.01.2026).
- [15] M. Polese, L. Bonati, S. D’Oro, S. Basagni, and T. Melodia, “CoO-RAN: Developing Machine Learning-Based xApps for Open RAN Closed-Loop Control on Programmable Experimental Platforms,” *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, pp. 5787–5800, Oct. 2023.
- [16] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing Atari with Deep Reinforcement Learning,” arXiv, cs.LG 1312.5602, Dec. 2013.
- [17] H. van Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-learning,” in *AAAI 2016*, Phoenix, AZ: AAAI Press, Feb. 2016, pp. 2094–2100.
- [18] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine Learning*, vol. 8, no. 3–4, pp. 229–256, May 1992.