Energy-based Surprise Minimization for Multi-Agent Value Factorization

Karush Suri  
CMTE, University of Toronto  
karush.suri@mail.utoronto.ca

Konstantinos Plataniotis  
Multimedia Laboratory, University of Toronto  
kostas@ece.utoronto.ca

Xiao Qi Shi  
RBC Capital Markets  
xiaopi.shi@borealisai.com

Yuri Lawryshyn  
CMTE, University of Toronto  
yuri.lawryshyn@utoronto.ca

ABSTRACT
Multi-Agent Reinforcement Learning (MARL) has demonstrated significant success in training decentralised policies in a centralised manner by making use of value factorization methods. However, addressing surprise across spurious states and approximation bias remain open problems for multi-agent settings. We introduce the Energy-based MIXer (EMIX), an algorithm which minimizes surprise utilizing the energy across agents. Our contributions are threefold: (1) EMIX introduces a novel surprise minimization technique across multiple agents in the case of multi-agent partially-observable settings. (2) EMIX highlights the first practical use of energy functions in MARL (to our knowledge) with theoretical guarantees and experiment validations of the energy operator. Lastly, (3) EMIX presents a novel technique for addressing overestimation bias across agents in MARL. When evaluated on a range of challenging StarCraft II micromanagement scenarios, EMIX demonstrates consistent state-of-the-art performance for multi-agent surprise minimization. Moreover, our ablation study highlights the necessity of the energy-based scheme and the need for elimination of overestimation bias in MARL. Our implementation of EMIX and videos of agents are available at https://karush17.github.io/emix-web/.

KEYWORDS
Value Factorization, Energy, Multi-Agent, EMIX, Surprise.

1 INTRODUCTION
Reinforcement Learning (RL) has seen tremendous growth in applications such as arcade games [37], board games [46, 49], robot control tasks [29, 48] and lately, real-time games [59]. The rise of RL has led to an increasing interest in the study of multi-agent systems [30, 58], commonly known as Multi-Agent Reinforcement Learning (MARL). In the case of partially observable settings, MARL enables the learning of policies with centralised training and decentralised control [22]. This has proven to be useful for exploiting value-based methods which are often found to be sample-inefficient [9, 54].

Value Factorization [42, 52] is a common technique which enables the joint value function to be represented as a combination of individual value functions conditioned on states and actions. In the case of Value Decomposition Network (VDN) [52], a linear additive factorization is carried out whereas QMIX [42] generalizes the factorization to a non-linear combination, hence improving the expressive power of centralised action-value functions. Furthermore, monotonicity constraints in QMIX enable scalability in the number of agents. On the other hand, factorization across multiple value functions leads to the aggregation of approximation biases [18, 19] originating from overoptimistic estimations in action values [11, 24] which remain an open problem in the case of multi-agent settings. Moreover, value factorization methods are conditioned on states and do not account for spurious changes in partially-observed observations, commonly referred to as surprise [1].

Surprise minimization [4] is a recent phenomenon observed in the case of single-agent RL methods which deals with environments consisting of spurious states. In the case of model-based RL [21], surprise minimization is used as an effective planning tool in the agent’s model [4] whereas in the case of model-free RL, surprise minimization is witnessed as an intrinsic motivation [1, 33] or generalization problem [8]. On the other hand, MARL does not account for surprise across agents as a result of which agents remain unaware of drastic changes in the environment [32]. Thus, surprise minimization in the case of multi-agent settings requires attention from a critical standpoint.

We introduce the Energy-based MIXer (EMIX), an algorithm based on QMIX which minimizes surprise utilizing the energy across agents. Our contributions are threefold: (1) EMIX introduces a novel surprise minimization technique across multiple agents in the case of multi-agent partially-observable settings. (2) EMIX highlights the first practical use of energy functions in MARL (to our knowledge) with theoretical guarantees and experiment validations of the energy operator. Lastly, (3) EMIX presents a novel technique for addressing overestimation bias across agents in MARL which, unlike previous single-agent methods [24], do not rely on a computationally-expensive family of action value functions. When evaluated on a range of challenging StarCraft II scenarios [45], EMIX demonstrates state-of-the-art performance for multi-agent surprise minimization by significantly improving the consistent performance of QMIX. Moreover, our ablation study highlights the necessity of our energy-based scheme and the need for elimination of overestimation bias in MARL.
2 THE VALUE FACTORIZATION PROBLEM

2.1 Preliminaries

We review the cooperative MARL setup. The problem is modeled as a Partially Observable Markov Decision Process (POMDP) [53] defined by the tuple \((S, \mathcal{A}, r, N, P, Z, O, \gamma)\) where the state space \(S\) and action space \(\mathcal{A}\) are discrete, \(r: S \times \mathcal{A} \rightarrow [r_{\min}, r_{\max}]\) presents the reward observed by agents \(a \in N\) where \(N\) is the set of all agents, \(P: \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, \infty)\) presents the unknown transition model consisting of the transition probability to the next state \(s' \in \mathcal{S}\) given the current state \(s \in \mathcal{S}\) and joint action \(u \in \mathcal{A}\) at time step \(t\) and \(\gamma\) is the discount factor. We consider a partially observable setting in which each agent \(n\) draws individual observations \(z \in Z\) according to the observation function \(O(s, a): \mathcal{S} \times \mathcal{A} \rightarrow Z\). We consider a joint policy \(\pi_g(u|z)\) as a function of model parameters \(\theta\). Standard RL defines the agent’s objective to maximize the expected discounted reward \(\mathbb{E}_\pi_0 \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, u_t)\right]\) as a function of the parameters \(\theta\). The action-value function for an agent is represented as \(Q(\pi_g(u|z|s, \theta) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, u_t)\right]\) which is the expected sum of payoffs obtained in state \(s\) upon performing action \(u\) by following the policy \(\pi_g\). We denote the optimal policy \(\pi^*_g\) such that \(Q(u, s; \theta^*) \geq Q(u, s; \theta)^*\) for \(s, u \in S, A\). In the case of multiple agents, the joint optimal policy can be expressed as the Nash Equilibrium [38] of the Stochastic Markov Game as \(\pi^* = (\pi_1^*, \pi_2^*, ..., \pi_n^*)\) such that \(Q(u^a, s; \theta^*) \geq Q(u^a, \theta^*)^*\) for \(s, u \in S, a \in A\). Q-Learning is an off-policy, model-free algorithm suitable for continuous and episodic tasks. The algorithm uses semi-gradient descent to minimize the Temporal Difference (TD) error: \(L(\theta) = \mathbb{E}_{b \sim R} [(y - Q(u, s; \theta))^2]\) where \(y = r + \gamma \max_{u' \in A'} Q(u', s'; \theta^*)\) is the TD target consisting of \(\theta^*\) as the target parameters and \(b\) is the batch sampled from memory \(R\).

2.2 Surprise Minimization

Despite the recent success of value-based methods [20, 36] RL agents suffer from spurious state spaces and encounter sudden changes in trajectories. These anomalous transitions between consecutive states are termed as surprise [1]. Quantitatively, surprise can be inferred as a measure of deviation [4, 8] among states encountered by the agent during its interaction with the environment. While exploring [5, 56] the environment, agents tend to have higher deviation among states which is gradually reduced by gaining a significant understanding of state-action transitions. Agents can then start selecting optimal actions which is essential for maximizing reward. These actions often lead the agent to spurious experiences which the agent may not have encountered. In the case of model-based RL, agents can leverage spurious experiences [4] and plan effectively for future steps. On the other hand, in the case of model-free RL, surprise results in sample-inefficient learning [1]. This can be tackled by making use of rigorous exploration strategies [27, 50]. However, such techniques do not necessarily scale to high-dimensional tasks and often require extrinsic feature engineering [23] and meta models [13]. A suitable way to tackle high-dimensional dynamics is by utilizing surprise as a penalty on the reward [8]. This leads to improved generalization. However, such solutions do not show evidence for multiple agents consisting of individual partial observations [43].

2.3 Overestimation Bias

Recent advances [11] in value-based methods have addressed overestimation bias (also known as approximation error) which stems from the value estimates approximated by the function approximator. Such methods make use of dual target functions [60] which improve stability in the Bellman updates. This has led to a significant improvement in single-agent off-policy RL methods [17]. However, MARL value-based methods continue to suffer from overestimation bias [2, 31]. Figure 1 highlights the overestimation bias originating from the overoptimistic estimations of the target value estimator. Plots present the variation of absolute TD error during learning for state-of-the-art MARL methods, namely Independent Q-Learning [54], Counterfactual Multi-Agent Policy Gradients (COMA) [9], VDN [52] and QMIX [42]. Significant rise in error values of value factorization methods such as QMIX and VDN presents the aggregation of errors from individual Q-value functions. Thus, overestimation bias in MARL value factorization requires attention from a critical standpoint.

![Figure 1: Absolute TD error for state-of-the-art MARL methods in StarCraft II micromanagement scenarios. Rise in error values depict the overoptimistic approximations estimated by the target value estimator.](image-url)
2.4 Energy-based Models

Energy-Based Models (EBMs) [25, 26] have been successfully applied in the field of machine learning [55] and probabilistic inference [34]. A typical EBM $E$ formulates the equilibrium probabilities \[^4\] $\Pr (v, h) = \frac{\exp (-E(v, h))}{\sum_{\hat{v}, \hat{h}} \exp (-E(\hat{v}, \hat{h}))}$ via a Boltzmann distribution [28] where $v$ and $h$ are the values of the visible and hidden variables and $\hat{v}$ and $\hat{h}$ are all the possible configurations of the visible and hidden variables respectively. The probability distribution over all the visible variables can be obtained by summing over all possible configurations of the hidden variables. This is mathematically expressed in Equation 1.

$$P(v) = \frac{\sum_{h} \exp (-E(v, h))}{\sum_{\hat{v}, \hat{h}} \exp (-E(\hat{v}, \hat{h}))}$$ (1)

Here, $E(v, h)$ is called the equilibrium free energy which is the minimum of the variational free energy and $\sum_{\hat{v}, \hat{h}} \exp (-E(\hat{v}, \hat{h}))$ is the partition function.

EBMs have been successfully implemented in single-agent RL methods [16, 40]. These typically make use of Boltzmann distributions to approximate policies [28]. Such a formulation results in the minimization of free energy within the agent. While policy approximation depicts promise in the case of unknown dynamics, inference methods [57] play a key role in optimizing goal-oriented behavior. A second type of usage of EBMs follows the maximization of entropy [65]. The maximum entropy framework [17] highlighted in Soft Q-Learning (SQL) [16] allows the agent to obey a policy which maximizes its reward and entropy concurrently. Application of agent’s entropy results in diverse and adaptive behaviors [64] which may be difficult to accomplish using standard exploration techniques [5, 56]. Moreover, the maximum entropy framework is equivalent to approximate inference in the case of policy gradient methods [47]. Such a connection between likelihood ratio gradient techniques and energy-based formulations leads to diverse and robust policies [14] and their hierarchical extensions [15] which preserve the lower levels of hierarchies.

In the case of MARL, EBMs have witnessed limited applicability as a result of the increasing number of agents and complexity within each agent [7]. While the probabilistic framework is readily transferable to opponent-aware multi-agent systems [62], cooperative settings consisting of coordination between agents require a firm formulation of energy which is scalable in the number of agents [12] and accounts for environments consisting of spurious states [61].

3 ENERGY-BASED SURPRISE MINIMIZATION

In this section, we introduce the novel surprise minimizing EMIX agent. The motivation behind EMIX stems from spurious states and overestimation bias among agents in the case of partially-observed settings. EMIX aims to address these challenges by making use of an energy-based surprise value function in conjunction with dual target function approximators.

3.1 The Surprise Minimization Objective

Firstly, we formulate the energy-based objective consisting of surprise as a function of states $s$, joint actions $u$ and deviation $\sigma$ within states for each agent $a$. We call this function as the surprise value function $\mathcal{V}^a_{\text{surp}}(s, u, \sigma)$ which serves as a mapping from agent and environment dynamics to surprise. We then define an energy operator presented in Equation 2 which sums the free energy across all agents.

$$\mathcal{V}^a_{\text{surp}}(s, u, \sigma) = \log \sum_{a=1}^{N} \exp (\mathcal{V}^a_{\text{surp}}(s, u, \sigma))$$ (2)

We make use of the Mellowmax operator [3] as our energy operator. The energy operator is similar to the SQL energy formulation [16] where the energy across different actions is evaluated. In our case, inference is carried out across all agents with actions as prior variables. However, in the special case of using an EBM as a Q-function, the EMIX objective reduces to the SQL objective. Details on connection between SQL and our energy formulation can be found in section 5.

Our choice of the energy operator is based on its unique mathematical properties which result in better convergence. Of these properties, the most useful result is that the energy operator forms a contraction on the surprise value function indicating a guaranteed minimization of surprise within agents. This is formally stated in Theorem 1. Proof of Theorem 1 can be found in section 5.

**Theorem 1.** Given a surprise value function $\mathcal{V}^a_{\text{surp}}(s, u, \sigma)$ of $a \in N$, the energy operator $\mathcal{V}^a_{\text{surp}}(s, u, \sigma) = \log \sum_{a=1}^{N} \exp (\mathcal{V}^a_{\text{surp}}(s, u, \sigma))$ forms a contraction on $\mathcal{V}^a_{\text{surp}}(s, u, \sigma)$.

The energy-based surprise minimization objective can then be formulated by simply adding the approximated energy-based surprise to the initial Bellman objective as expressed below.

$$L(\theta) = \mathbb{E}_{b \sim R_b} \left[ \frac{1}{2} (y - (Q(u, s; \theta) + \beta \log \sum_{a=1}^{N} \exp (\mathcal{V}^a_{\text{surp}}(s, u, \sigma)))^2 \right]$$

$$= \mathbb{E}_{b \sim R_b} \left[ \frac{1}{2} (r + \gamma \max_{u'} Q(u', s'; \theta^-) + \beta \log \sum_{a=1}^{N} \exp (\mathcal{V}^a_{\text{surp}}(s', u', \sigma')) - \gamma Q(u, s; \theta) + \beta \log \sum_{a=1}^{N} \exp (\mathcal{V}^a_{\text{surp}}(s, u, \sigma)))^2 \right]$$

$$= \mathbb{E}_{b \sim R_b} \left[ \frac{1}{2} (r + \gamma \max_{u'} Q(u', s'; \theta^-) + \beta \log \frac{\sum_{a=1}^{N} \exp (\mathcal{V}^a_{\text{surp}}(s', u', \sigma'))}{\sum_{a=1}^{N} \exp (\mathcal{V}^a_{\text{surp}}(s, u, \sigma)))} - \gamma Q(u, s; \theta))^2 \right]$$

$$L(\theta) = \mathbb{E}_{b \sim R_b} \left[ \frac{1}{2} (r + \gamma \max_{u'} Q(u', s'; \theta^-) + \beta E - \gamma Q(u, s; \theta))^2 \right]$$ (3)

Here, $E$ is defined as the surprise ratio with $\beta$ as a temperature parameter and $\sigma'$ as the deviation among next states in the batch. The surprise value function is approximated by a universal function approximator with its parameters as $\phi$. $E$ is expressed as the surprise energy function with $\mathcal{V}^a(s', u', \sigma')$ being the negative free energy and $\sum_{a=1}^{N} \exp (\mathcal{V}^a(s, u, \sigma))$ the partition function. Alternatively, $\mathcal{V}^a(s, u, \sigma)$ can be formulated as the negative free energy with $\sum_{a=1}^{N} \exp (\mathcal{V}^a(s, u, \sigma'))$ as the partition function. The objective incorporates the minimization of surprise across all agents.
as minimizing the energy in spurious states. Such a formulation of surprise acts as intrinsic motivation and at the same time provides robustness to multi-agent behavior. Furthermore, the energy formulation in the form of energy ratio $E$ is a suitable one as it guarantees convergence to minimum surprise at optimal policy $\pi^*$. This is formally expressed in Theorem 2 with its corresponding proof in section 5.

**Theorem 2.** Upon agent’s convergence to an optimal policy $\pi^*$, total energy of $\pi^*$, expressed by $E^*$ will reach a thermal equilibrium consisting of minimum surprise among consecutive states $s$ and $s'$.

The objective can be modified to tackle approximation error in the target $Q$-values. We introduce a total of $m$ target approximators making $\{Q_1(u, s; \theta^-), Q_2(u, s; \theta^-), ..., Q_m(u, s; \theta^-)\}$ as the set of target approximators. However, unlike generalized $Q$-learning [24], we do not instantiate another $Q$-function but simply keep a copy of $\theta$ and select the target estimates with minimum values during optimization. This allows the objective to address overestimation bias in a scalable manner without using multiple $Q$-functions. The final EMIX objective is mathematically expressed in Equation 4.

$$L(\theta) = \mathbb{E}_{(u,r,s') \sim \mathcal{R}} \left[ \frac{1}{|\mathcal{R}|} \frac{1}{2 \alpha} \left( (r + \gamma \max_{\theta} Q_i(u, s; \theta^-) + \beta E - Q(u, s; \theta)) \right)^2 \right]$$ \(4\)

Here, $i$ depicts each of the $m$ target estimators with $\min_i Q_i(u', s'; \theta^-)$ indicating the estimate with minimum error.

### 3.2 Energy-based MIXer (EMIX)

**Algorithm 1** Energy-based MIXer (EMIX)

1. Initialize $\phi$, $\theta$, $\theta_1^-$, ..., $\theta_m^-$, agent and hypernetwork parameters.
2. Initialize learning rate $\alpha$, temperature $\beta$ and replay buffer $\mathcal{R}$.
3. for environment step do
4. $u \leftarrow (u_1, u_2, ..., u_N)$
5. $\mathcal{R} \leftarrow \mathcal{R} \cup \{(s, u, r, s')\}$
6. if $|\mathcal{R}| >$ batch-size then
7. for random batch do
8. $Q_{tot}^\alpha \leftarrow$ Mixer-Network($Q_1, Q_2, ..., Q_N, s$)
9. $Q_i^\alpha \leftarrow$ Target-Mixer($Q_1, Q_2, ..., Q_N, s$), $\forall i = 1,2,..,m$
10. Calculate $\sigma$ and $\sigma'$ using $s$ and $s'$
11. $V_{surp}^\alpha(s, u, \sigma) \leftarrow$ Surprise-Mixer($s, u, \sigma$)
12. $V_{surp}^\alpha(s', u, \sigma') \leftarrow$ Target-Surprise-Mixer($s', u, \sigma'$)
13. $E \leftarrow \log \sum_{i=1}^{N} \exp(V_{surp}^\alpha(s, u, \sigma))$
14. Calculate $L(\theta)$ using $E$ in Equation 4
15. $\theta \leftarrow \theta - \alpha \nabla_{\theta} L(\theta)$
16. end for
17. end if
18. if update-interval steps have passed then
19. $\theta_i^- \leftarrow \theta, \forall i = 1,2,..,m$
20. end if
21. end for

Algorithm 1 presents the EMIX algorithm. We initialize surprise value function parameters $\phi$, mixer parameters $\theta$, target parameters $\theta_i^-$ for $i = 1,2,..,m$ and lastly the agent and hypernetwork parameters of QMIX. A learning rate $\alpha$, temperature $\beta$ and replay buffer $\mathcal{R}$ are instantiated. During environment interactions, agents in state $s$ perform joint action $u$, observe reward $r$ and transition to next-states $s'$. These experiences are collected in $\mathcal{R}$ as $(s, u, r, s')$ tuples. In order to make the agents explore the environment, an $\epsilon$-greedy schedule is used similar to the original QMIX [42] implementation. During the update steps, a random batch of batch – size is sampled from $\mathcal{R}$. The total $Q$-value $Q_{tot}^\alpha$ is computed by the mixer network with its inputs as the $Q$-values of all the agents conditioned on $s$ via the hypernetworks. Similarly, the target mixers approximate $Q_i^\alpha$ conditioned on $s'$. In order to evaluate surprise within agents, we compute the standard deviations $\sigma$ and $\sigma'$ across all observations $z$ and $z'$ for each agent using $s$ and $s'$ respectively. The surprise value function called the Surprise-Mixer estimates the surprise $V_{surp}(s, u, \sigma)$ conditioned on $s$, $u$ and $\sigma$. The same computation is repeated using the Target-Surprise-Mixer for estimating surprise $V_{surp}^\alpha(s, u, \sigma)$ within next-states in the batch. Application of the energy operator along the non-singleton agent dimension for $V_{surp}^\alpha(s, u, \sigma)$ and $V_{surp}^\alpha(s', u, \sigma')$ yields the energy ratio $E$ which is used in Equation 4 to evaluate $L(\theta)$. We then use batch gradient descent to update parameters of the mixer $\theta$. Target parameters $\theta_i^-$ are updated every update – interval steps.

We now take a closer look at the surprise-mixer approximating the surprise value function. In order to condition surprise on states, joint actions and the deviation among states, we construct an expressive architecture motivated by provable exploration in RL [35]. The original architecture constructs a state abstraction model for a classification setting. It maps the transitions consisting of states $s$, actions $a$ and next-states $s'$ to the conditional probability $p(y|s, a, s')$ depicting whether the transition belongs to the same data distribution $y$ or not. Such models have proven to be efficient in the case of provable exploration [35] as it allows the agent to learn an exploration policy for every value of abstract state related to the latent space. We borrow from this technique of provable exploration and extend it to the surprise minimization setting.

![Figure 2: Surprise-Mixer architecture for estimation of the surprise value function.](Image 325x119 to 551x289)
Figure 2 presents the expressive architecture of surprise-mixer network utilized for surprise value function approximation and minimization. In contrast to the original state abstraction model [35], the surprise-mixer maps transitions consisting of states $s$, joint actions $u$ and deviations $\sigma$ to a surprise value $V_{\text{surp}}^a(s, u, \sigma)$ for all agents $a$. Hierarchical layers of the network aid in the extraction of latent space representations folowed by the estimation of $V_{\text{surp}}^a(s, u, \sigma)$. The architecture allows the agent to learn a robust and surprise-agnostic policy for every value of abstract state related to the latent space. Moreover, the latent space accommodates every value of surprise across agents as a result of state deviations induced in the intermediate representations. We refrain from passing next-states $s'$ as part of the transitions in order to maintain causality in the system. Surprise value estimates $V_{\text{surp}}^a(s, u, \sigma)$ are evaluated by the energy operator with the resulting expression becoming a part of the Bellman objective in Equation 4 comprising of the total $Q$-values $Q_{\text{tot}}$ estimated by the mixer network.

4 EXPERIMENTS

Our experiments aim to evaluate the performance, consistency, sample-efficiency and effectiveness of the various components of our method. Specifically, we aim to answer the following questions: (1) How does our method compare to current state-of-the-art MARL methods in terms of performance, consistency and sample-efficiency?, (2) How much does each component of the method contribute to its performance? and (3) Does the algorithm validate the theoretical claims corresponding to its components?

4.1 Energy-based Surprise Minimization

We assess the performance and sample-efficiency of EMIX on multi-agent StarCraft II micromanagement scenarios [45]. We select StarCraft II scenarios particularly for three reasons. Firstly, micromanagement scenarios consist of a larger number of agents with different action spaces. This requires a greater deal of coordination in comparison to other benchmarks [51] which attend to other aspects of MARL performance such as opponent-awareness [6]. Secondly, micromanagement scenarios consist of partial observability wherein agents are restricted from responding to enemy fire and attacking enemies when they are in range [42]. This allows agents to explore the environment effectively and find an optimal strategy purely based on collaboration rather than built-in game utilities. Lastly, micromanagement scenarios in StarCraft II consist of multiple opponents which introduce a greater degree of surprise within consecutive states. Irrespective of the time evolution of an episode, environment dynamics of each scenario change rapidly as the agents need to respond to enemy’s behavior.

We compare our method to current state-of-the-art methods, namely QMIX [42], VDN [52], COMA [9] and IQL [54]. In order to compare our surprise-based scheme against pre-existing surprise minimization mechanisms, we compare EMIX additionally to a model-free implementation of SMiRL [4] in QMIX. All methods were implemented using the PyMARL framework [45]. The SMiRL component was additionally incorporated as per the update rule provided in [8]. We use the generalized version of SMiRL as it demonstrates reduced variance across batches. We term this implementation as SMiRL-QMIX for our comparisons. Agents were trained for a total of 5 random seeds consisting of 2 million steps in each environment. A total of 32 validation episodes carried out at every 10,000 step intervals were interleaved during agent’s interactions. All baselines implementation consist of a Recurrent Neural Network (RNN) agent having memory consisting of past states and actions. We use an $\epsilon$-greedy exploration scheme wherein $\epsilon$ is annealed from 1 to 0.01 during the initial stages of training. Details related to the implementation of EMIX are presented in section 5.

In order to assess the performance and sample-efficiency of agents we evaluate the success rate percentages of each multi-agent system in completing each scenario. A completion of a scenario indicates the victory of the team over its enemies. Scenarios consist of varying difficulties in terms of the number of agents, map locations, distance from enemies, number of enemies and the level of difficulty.

Table 1 presents the comparison of success rate percentages between EMIX and state-of-the-art MARL algorithms on the StarCraft II micromanagement scenarios. Along with the success rates, we also measure the deviation of performance across the 5 random seeds considered during experiments. Complete results for all scenarios including plots presenting agents’ learning behaviors can be viewed in section 5. We evaluate the performance of agents on a total of 12 scenarios. Naming conventions of the scenarios are in accordance with the multi-agent StarCraft II micromanagement framework [45] wherein $s$ represents an agent belonging to the stalker unit, $z$ signifies a zealot unit, $m$ indicates a marine unit and $sc$ implies a spine crawler unit. Corresponding to each scenario, algorithms demonstrating higher success rate values in comparison to other methods have their entries highlighted. Out of the 12 scenarios considered, EMIX presents higher success rates on 9 of these scenarios depicting the suitability of the proposed approach.

In scenarios such as 3m, 3s5z and 8m performance gain between EMIX and other methods such as QMIX and VDN are incremental as a result of the small number of agents and simplicity of tasks. On the other hand, EMIX presents significant performance gains in cases of so_many_baneling and 5m_vs_6m which consist of a large number of opponents and a greater difficulty level respectively.

When compared to QMIX, EMIX depicts improved success rates on all of the 12 scenarios. For instance, in scenarios such as 3s_vs_5z, 8m_vs_9m and 5m_vs_6m QMIX presents sub-optimal performance. On the other hand, EMIX utilizes a comparatively improved joint policy and yields better convergence in a sample-efficient manner. Thus, EMIX augments the performance and sample-efficiency of the QMIX agent utilizing the energy-based surprise minimization scheme. Moreover, on comparing EMIX with SMiRL-QMIX, we note that EMIX demonstrates a higher average success rate. This highlights the suitability of the energy-based scheme in the case of a larger number of agents and complex environment dynamics for surprise minimization.

In addition to state-of-the-art performance and sample-efficiency, EMIX also presents consistency in its learning across different random seeds. Deviation in success rates for EMIX is comparable to pre-existing value factorization methods such as QMIX and VDN. This indicates that the energy-based formulation of surprise minimization is compatible with value factorization and enables all the agents to exhibit the same optimal behavior across different runs thus, validating the suitability of the proposed approach.
We now present the ablation study for the various components we remove the energy-based surprise minimization from EMIX and Table 1: Comparison of success rate percentages between EMIX and state-of-the-art MARL methods for StarCraft II micro-management scenarios. Results are averaged over 5 random seeds with each session consisting of 2 million environment interactions. EMIX significantly improves the performance of the QMIX agent on a total of 9 out of 12 scenarios. In addition, EMIX presents less deviation between its random seeds indicating consistency in collaboration across agents.

### 4.2 Ablation Study

We now present the ablation study for the various components of EMIX. Our experiments aim to determine the effectiveness of the energy-based surprise minimization method and the multiple target $Q$-function scheme. Additionally, we also aim to determine the extent up to which our proposed framework is viable in the standard QMIX objective.

#### 4.2.1 Energy-based Surprise Minimization and Overestimation Bias

To weigh the effectiveness of the multiple target $Q$-function scheme we remove the energy-based surprise minimization from EMIX and replace it with the prior QMIX objective. For simplicity, we make use of two target $Q$-functions. We call this implementation of QMIX combined with the dual target function scheme as TwinQMIX. We can now add the energy-based surprise minimization scheme in the TwinQMIX objective to retrieve the EMIX objective. Thus, we can compare between QMIX, TwinQMIX and EMIX to assess the contributions of each of the proposed methods. Figure 3 (top) presents the comparison of average success rates for QMIX, TwinQMIX and EMIX on six different scenarios. Agents were evaluated for a total of 2 million timesteps with the lines in the plot indicating average success rates and the shaded area as the deviation across 5 random seeds.

In comparison to QMIX, TwinQMIX adds stability to the original objective and yields performance gains in the form of improved success rates and sample-efficient convergence. For instance, in the $3s\_vs\_5z$ scenario, TwinQMIX significantly improves the performance of QMIX by reducing the overoptimistic estimates in the initial QMIX objective. However, in the $5m\_vs\_6m$ scenario, TwinQMIX falls short of optimal sample efficiency as a result of underestimative estimates yielded by the $\min Q'_s(s, u, \sigma)$ operation.

On comparing TwinQMIX to EMIX we note that the energy-based surprise minimization scheme provides significant performance improvement in the modified QMIX objective. The EMIX objective demonstrates sample-efficiency and greater success rate values when compared to the TwinQMIX implementation. Additionally, the surprise minimization term $\beta E$ adds to the stability of the TwinQMIX objective. This is demonstrated in the $5m\_vs\_6m$ scenario wherein the EMIX implementation improves the performance of TwinQMIX in comparison to QMIX by compensating for the underoptimistic estimations in the bellman updates. In the case of so_many_baneling scenario, EMIX tackles surprise effectively by preventing a significant drop in performance which is observed in cases of QMIX and TwinQMIX. so_many_baneling scenario consists of a large number of opponents (27 banelings) which force the agents to act quickly. This inherently induces a large amount of surprise in the form of state-to-state deviations. EMIX successfully tackles this hindrance and prevents the drop in success rates as a result of a surprise-robust policy.

#### 4.2.2 Temperature Parameter

We now evaluate the extent of effectiveness of our surprise minimization objective in accordance with the temperature parameter $\beta$. Figure 3 (middle) presents the variation of success rates of the EMIX objective with $\beta$ during learning. EMIX was evaluated for three different values (as presented in the legend) of $\beta$ for a total of 5 random seeds. While the objective is robust to significant changes in the value of $\beta$, it presents sub-optimal performance in the case of high ($\beta = 0.1$) and low ($\beta = 0.001$) temperature values. In the case of high $\beta$ values, the objective suffers from overestimation error in the bellman updates introduced by the energy term. The error compensates for the bias removed by the dual $Q$-function scheme. On the other hand, low $\beta$ values do not include surprise minimization and EMIX agents face spurious states as a result of negligible surprise minimization. For instance, $5m\_vs\_6m$ and $8m\_vs\_9m$ scenarios highlight the necessity for a suitably high $\beta$ in order to balance the surprise minimization objective with the initial bellman updates.

The importance of $\beta$ can be validated by assessing its usage in surprise minimization. However, it is difficult evaluate surprise minimization directly as surprise value function estimates $V^a_{\text{surp}}(s, u, \sigma)$ vary from state-to-state across different agents and thus, they
Figure 3: Ablations on six different scenarios for each of EMIX’s components (top), its variation in performance (middle) and surprise minimization (bottom) with temperature $\beta$. When compared to QMIX, EMIX and TwinQMIX (EMIX without surprise minimization) depict improved performance and sample efficiency indicating the suitability of energy-based surprise minimization and dual Q-function scheme. This is achieved by making use of a suitable value of temperature parameter ($\beta = 0.01$) which controls the extent of energy ratio $E$ in the EMIX objective. Moreover, the temperature parameter affects the stability in surprise minimization by utilizing $E$ as intrinsic motivation.

5 CONCLUSION

In this paper, we introduced the Energy-based MIXer (EMIX), a multi-agent value factorization algorithm based on QMIX which minimizes surprise utilizing the energy across agents. Our method proposes a novel energy-based surprise minimization objective consisting of an energy operator in conjunction with the surprise value function across multiple agents in the case of multi-agent partially-observable settings. The EMIX objective satisfies theoretical guarantees of total energy and surprise minimization with experimental results validating these claims. Additionally, EMIX presents a novel technique for addressing overestimation bias across agents in MARL based on multiple target value approximators. Unlike previous single-agent methods, EMIX does not rely on a computationally-expensive family of action value functions. On a range of challenging StarCraft II micromanagement scenarios, EMIX demonstrates state-of-the-art performance and sample-efficiency for multi-agent surprise minimization by significantly improving the original QMIX objective. Our ablations carried out on the proposed energy-based scheme, multiple target approximators and temperature parameter highlight the suitability and significance of each of the proposed contributions. While EMIX serves as the first practical example (to our knowledge) of energy-based models in cooperative MARL, we aim to extend the energy framework to opponent-aware and hierarchical MARL. We leave this as our future work.

LINK TO FULL APPENDIX

The full appendix consisting of complete details of the study can be found at https://www.dropbox.com/Appendix.pdf.
ACKNOWLEDGMENTS
We would like to thank the anonymous reviewers for providing valuable feedback on our work. We acknowledge Aravind Varier and Shashank Saurav for helpful discussions and the computing platform provided by the Department of Computer Science (DCS), University of Toronto. This work is supported by RBC Capital Markets, RBC Innovation Lab and the Center for Management of Technology and Entrepreneurship (CMTE).

ABRIDGED APPENDIX
Proofs
Theorem 1. Let us first define a norm on surprise values \( ||V_1 - V_2|| \equiv \max_{s, u, \sigma} |V_1(s, u, \sigma) - V_2(s, u, \sigma)| \). Suppose \( \epsilon = ||V_1 - V_2|| \).

\[
\log \sum_{a=1}^{N} \exp (V_1(s, u, \sigma)) \leq \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma) + \epsilon)
\]

\[
= \log \sum_{a=1}^{N} \exp (V_1(s, u, \sigma)) \leq \exp (\epsilon) \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma))
\]

\[
= \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma)) \leq \epsilon + \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma))
\]

\[
= \log \sum_{a=1}^{N} \exp (V_1(s, u, \sigma)) - \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma)) \leq ||V_1 - V_2||
\]

(5)

Similarly, using \( \epsilon \) with log- \( \sum_{a=1}^{N} \exp (V_1(s, u, \sigma)) \).

\[
\log \sum_{a=1}^{N} \exp (V_1(s, u, \sigma) + \epsilon) \leq \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma))
\]

\[
= \log \exp (\epsilon) \sum_{a=1}^{N} \exp (V_1(s, u, \sigma)) \leq \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma))
\]

\[
= \epsilon + \log \sum_{a=1}^{N} \exp (V_1(s, u, \sigma)) \leq \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma))
\]

\[
= ||V_1 - V_2|| \geq \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma)) - \log \sum_{a=1}^{N} \exp (V_1(s, u, \sigma))
\]

(6)

Results in Equation 5 and Equation 6 prove that the energy operation is a contraction.

Theorem 2. We begin by initializing a set of \( M \) policies \( \{ \pi_1, \pi_2, ..., \pi_M \} \) having energy ratios \( E_1, E_2, ..., E_M \). Consider a policy \( \pi_1 \) with surprise value function \( V_1 \). \( E_1 \) can then be expressed as

\[
E_1 = \log \left[ \frac{\sum_{a=1}^{N} \exp (V_1^a(s, u, \sigma) + \xi_1)}{\sum_{a=1}^{N} \exp (V_1^a(s, u, \sigma))} \right]
\]

Assuming a constant surprise between \( s \) and \( s' \), we can express \( V_1^a(s', u', \sigma') = V_1^a(s, u, \sigma) + \xi_1 \) where \( \xi_1 \) is a constant. Using this expression in \( E_1 \) we get,

\[
E_1 = \log \left[ \frac{\sum_{a=1}^{N} \exp (V_1^a(s', u', \sigma') + \xi_1)}{\sum_{a=1}^{N} \exp (V_1^a(s, u, \sigma))} \right]
\]

Similarly, \( E_2 = \xi_2, E_3 = \xi_3, ..., E_M = \xi_M \). Thus, the energy residing in policy \( \pi \) is proportional to the surprise between consecutive states \( s \) and \( s' \). Clearly, an optimal policy \( \pi^* \) is the one with minimum surprise. Mathematically,

\[
\pi^* \geq \pi_1, \pi_2, ..., \pi_M \implies \xi^* \leq \xi_1, \xi_2, ..., \xi_M
\]

\[
= \pi^* \geq \pi_1, \pi_2, ..., \pi_M \implies E^* \leq E_1, E_2, ..., E_M
\]

Thus, proving that the optimal policy consists of minimum surprise at thermal equilibrium.

Implementation Details
Model Specifications. This section highlights model architecture for the surprise value function. At the lower level, the architecture consists of 3 independent networks called state_net, q_net and surp_net. Each of these networks consist of a single layer of 256 units with ReLU non-linearity as activations. Similar to the mixer-network, we use the ReLU non-linearity in order to provide monotonicity constraints across agent. Using a modular architecture in combination with independent networks leads to a richer extraction of joint latent transition space. Outputs from each of the networks are concatenated and are provided as input to the main_net consisting of 256 units with ReLU activations. The main_net yields a single output as the surprise value \( V_{surp}^a(s, u, \sigma) \) which is reduced along the agent dimension by the energy operator. Alternatively, deeper versions of networks can be used in order to make the extracted embeddings increasingly expressive. However, increasing the number of layers does little in comparison to additional computational expense.

Hyperparameters. Table 2 presents hyperparameter values for EMIX. Value of \( \beta \) was tuned between 0.001 and 1 in intervals of 0.001 with best performance observed at \( \beta = 0.01 \). A total of 2 target Q-functions were used as the model is found to be robust to any greater values.

| Hyperparameters       | Values               |
|-----------------------|----------------------|
| batch size            | \( b = 32 \)         |
| learning rate         | \( \alpha = 0.0005 \) |
| discount factor       | \( \gamma = 0.99 \)   |
| target update interval| 200 episodes         |
| gradient clipping     | 10                   |
| exploration schedule  | 1.0 to 0.01 over 50000 steps |
| mixer embedding size  | 32                   |
| agent hidden size     | 64                   |
| temperature           | \( \beta = 0.01 \)    |
| target Q-functions    | 2                    |

Table 2: Hyperparameter values for EMIX agents
[53] Richard S. Sutton and Andrew G. Barto. 2018. Reinforcement Learning: An Introduction.

[54] Ming Tan. 1993. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In Proceedings of the Tenth International Conference on Machine Learning.

[55] Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E. Hinton. 2003. Energy-based models for sparse overcomplete representations. Journal of Machine Learning Research 4 (2003).

[56] Sebastian Thrun. 1992. Efficient exploration in reinforcement learning. (1992).

[57] Marc Toussaint. 2009. Robot trajectory optimization using approximate inference. In Proceedings of the 26th annual international conference on machine learning.

[58] Oriol Vinyals, Igor Babuschkin, Wojciech Czarnecki, Michal Misiak, Andrew Dudzik, Junyoung Chung, David Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John Agapiou, Max Jaderberg, and David Silver. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 575 (11 2019).

[59] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Allreza Makhzani, Heinrich Kajitani, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy Lillicrap, Kevin Cruce, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekeremo, Jacob Repp, and Rodney Tsing. 2016. StarCraft II: A New Challenge for Reinforcement Learning. arXiv:1708.04782

[60] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. 2016. Dueling network architectures for deep reinforcement learning. In International conference on machine learning.

[61] Ermo Wei, Drew Wicke, David Freelan, and Sean Luke. 2018. Multiagent soft q-learning. arXiv preprint arXiv:1804.09817 (2018).

[62] Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and Wei Pan. 2019. Probabilistic recursive reasoning for multi-agent reinforcement learning. arXiv preprint arXiv:1901.09207 (2019).

[63] Yan Zheng, Zhaopeng Meng, Jinye Hao, and Zonghang Zhang. 2018. Weighted double deep multiagent reinforcement learning in stochastic cooperative environments. In Pacific Rim international conference on artificial intelligence.

[64] Brian D Ziebart. 2010. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. (2010).

[65] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum entropy inverse reinforcement learning. In AAAI.
Supplementary Material- Energy-based Surprise Minimization for Multi-Agent Value Factorization

1 Proofs

Theorem 1. Given a surprise value function $V_{\text{surp}}^a(s, u, \sigma) \forall a \in N$, the energy operator $\log \sum_{a=1}^{N} \exp (V_{\text{surp}}^a(s, u, \sigma))$ forms a contraction on $V_{\text{surp}}^a(s, u, \sigma)$.

Proof. Let us first define a norm on surprise values $||V_1 - V_2|| \equiv \max_{s, u, \sigma} |V_1(s, u, \sigma) - V_2(s, u, \sigma)|$. Suppose $\epsilon = ||V_1 - V_2||$,

\[
\log \sum_{a=1}^{N} \exp (V_1(s, u, \sigma)) \leq \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma) + \epsilon)
\]
\[
= \log \sum_{a=1}^{N} \exp (V_1(s, u, \sigma)) \leq \log \exp (\epsilon) \sum_{a=1}^{N} \exp (V_2(s, u, \sigma))
\]
\[
= \log \sum_{a=1}^{N} \exp (V_1(s, u, \sigma)) \leq \epsilon + \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma))
\]
\[
= \log \sum_{a=1}^{N} \exp (V_1(s, u, \sigma)) - \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma)) \leq ||V_1 - V_2|| \quad (6)
\]

Similarly, using $\epsilon$ with $\log \sum_{a=1}^{N} \exp (V_1(s, u, \sigma))$,

\[
\log \sum_{a=1}^{N} \exp (V_1(s, u, \sigma) + \epsilon) \geq \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma))
\]
\[
= \log \exp (\epsilon) \sum_{a=1}^{N} \exp (V_1(s, u, \sigma)) \geq \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma))
\]
\[
= \epsilon + \log \sum_{a=1}^{N} \exp (V_1(s, u, \sigma)) \geq \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma))
\]
\[
= ||V_1 - V_2|| \geq \log \sum_{a=1}^{N} \exp (V_2(s, u, \sigma)) - \log \sum_{a=1}^{N} \exp (V_1(s, u, \sigma)) \quad (7)
\]

Results in Equation 6 and Equation 7 prove that the energy operation is a contraction. \qed
Theorem 2. Upon agent’s convergence to an optimal policy \( \pi^* \), total energy of \( \pi^* \), expressed by \( E^* \) will reach a thermal equilibrium consisting of minimum surprise among consecutive states \( s \) and \( s' \).

Proof. We begin by initializing a set of \( M \) policies \( \{ \pi_1, \pi_2, ..., \pi_M \} \) having energy ratios \( \{ E_1, E_2, ..., E_M \} \). Consider a policy \( \pi_1 \) with surprise value function \( V_1 \). \( E_1 \) can then be expressed as

\[
E_1 = \log \left( \frac{\sum_{a=1}^{N} \exp \left( V_1^a(s', u', \sigma') \right)}{\sum_{a=1}^{N} \exp \left( V_1^a(s, u, \sigma) \right)} \right)
\]

Assuming a constant surprise between \( s \) and \( s' \), we can express \( V_1^a(s', u', \sigma') = V_1^a(s, u, \sigma) + \zeta_1 \) where \( \zeta_1 \) is a constant. Using this expression in \( E_1 \) we get,

\[
E_1 = \log \left( \frac{\exp (\zeta_1) \sum_{a=1}^{N} \exp \left( V_1^a(s, u, \sigma) \right)}{\sum_{a=1}^{N} \exp \left( V_1^a(s, u, \sigma) \right)} \right) \cdot \zeta_1
\]

Similarly, \( E_2 = \zeta_2, E_3 = \zeta_3, ..., E_M = \zeta_M \). Thus, the energy residing in policy \( \pi \) is proportional to the surprise between consecutive states \( s \) and \( s' \). Clearly, an optimal policy \( \pi^* \) is the one with minimum surprise. Mathematically,

\[
\pi^* \geq \pi_1, \pi_2, ..., \pi_M \implies \zeta^* \leq \zeta_1, \zeta_2, ..., \zeta_M
\]

\[
\pi^* \geq \pi_1, \pi_2, ..., \pi_M \implies E^* \leq E_1, E_2, ..., E_M
\]

Thus, proving that the optimal policy consists of minimum surprise at thermal equilibrium.

2 Connection between EMIX and Soft Q-Learning

The Soft Q-Learning objective is given by-

\[
J_Q(\theta) = \mathbb{E}_{s,u \sim R} \left[ \frac{1}{2} \left( r + \gamma \mathbb{E}_{s' \sim R} [V_{soft}^\theta(s')] - Q_{soft}(u', s')] \right)^2 \right]
\]

\[
= J_Q(\theta) = \mathbb{E}_{s,u \sim R} \left[ \frac{1}{2} \left( r + \gamma \mathbb{E}_{s' \sim R} [\log \sum_{u \in A} \exp Q(u', s'; \theta^-)] - Q_{soft}(u', s') \right)^2 \right]
\]

The gradient of this objective can be expressed as-

\[
\nabla \theta J_Q(\theta) = \mathbb{E}_{s,u \sim R} \left[ (r + \gamma \mathbb{E}_{s' \sim R} [\log \sum_{u \in A} \exp Q(u', s'; \theta^-)] - Q_{soft}(u', s')) \nabla Q_{soft}(u, s) \right]
\]

(8)
And the gradient of the EMIX objective is obtained as-

\[
L(\theta) = \mathbb{E}_{s,u,s' \sim R} \left[ \frac{1}{2} (r + \gamma \max_{u'} \min_{i} Q_i(u', s'; \theta^-)) + \beta \log \left( \frac{\sum_{a=1}^{N} \exp \left( V_a^{\text{surp}}(s', u', \sigma') \right)}{\sum_{a=1}^{N} \exp \left( V_a^{\text{surp}}(s, u, \sigma) \right)} \right) - Q(u, s; \theta)^2 \right]
\]

\[
\nabla_{\theta} L(\theta) = \mathbb{E}_{s,u,s' \sim R} \left[ (r + \gamma \max_{u'} \min_{i} Q_i(u', s'; \theta^-)) + \beta \log \left( \frac{\sum_{a=1}^{N} \exp \left( V_a^{\text{surp}}(s', u', \sigma') \right)}{\sum_{a=1}^{N} \exp \left( V_a^{\text{surp}}(s, u, \sigma) \right)} \right) - Q(u, s; \theta) \right] \nabla_{\theta} Q(u, s; \theta) \quad (9)
\]

Comparing Equation 8 to Equation 9 we notice that Soft Q-Learning and EMIX are related to each other as they utilize energy-based models. Soft Q-Learning makes use of a discounted energy function which downweights the energy values over longer horizons. Actions consisting of lower energy configurations are given preference by making use of \(Q^{\text{soft}}(u, s; \theta)\) as the negative energy. On the other hand, EMIX makes use of a constant energy function weighed by \(\beta\) which minimizes surprise-based energy between consecutive states. Both the objectives can be thought of as energy minimizing models which search for an optimal energy configuration. Soft Q-Learning searches for an optimal configuration in the action space whereas EMIX favours optimal behavior on spurious states. In fact, EMIX can be realized as a special case of Soft Q-Learning if the mixer agent utilizes an energy-based policy and attains thermal equilibrium. This leads us to express Theorem 3.

**Theorem 3.** Given an energy-based policy \(\pi_{en}\) with its target function \(V(s') = \log \sum_{u \in A} \exp Q(u', s'; \theta^-)\), the surprise minimization objective \(L(\theta)\) reduces to the Soft Q-Learning objective \(L(\theta^{\text{soft}})\) in the special case when the variational free energy function \(\sum_{a=1}^{N} \exp \left( V_a^{\text{surp}}(s, u, \sigma) \right)\) is equal to the partition function \(\sum_{a=1}^{N} \exp \left( V_a^{\text{surp}}(s', u, \sigma') \right)\).

**Proof.** We know that the EMIX objective is given by-

\[
L(\theta) = \mathbb{E}_{s,u,s' \sim R} \left[ \frac{1}{2} (r + \gamma \max_{u'} \min_{i} Q_i(u', s'; \theta^-)) + \beta \log \left( \frac{\sum_{a=1}^{N} \exp \left( V_a^{\text{surp}}(s', u', \sigma') \right)}{\sum_{a=1}^{N} \exp \left( V_a^{\text{surp}}(s, u, \sigma) \right)} \right) - Q(u, s; \theta)^2 \right]
\]

Replacing the greedy policy term \(\max_{u'} \min_{i} Q_i(u', s'; \theta^-)\) with the energy-based...
target function $V(s') = \log \sum_{u' \in A} \exp Q(u', s'; \theta^{-})$, we get,

$$L(\theta) = \mathbb{E}_{s,s' \sim R}[\frac{1}{2}(r + \gamma \mathbb{E}_{s' \sim R}[V(s')])$$

$$+ \beta \log(\frac{\sum_{a=1}^{N} \exp (V_{surp}^{a}(s', u', \sigma'))}{\sum_{a=1}^{N} \exp (V_{surp}^{a}(s, u, \sigma))}) - Q(u, s; \theta)^2]$$

$$= L(\theta) = \mathbb{E}_{s,s' \sim R}[\frac{1}{2}(r + \gamma \mathbb{E}_{s' \sim R}[\log \sum_{u' \in A} \exp Q(u', s'; \theta^{-})])$$

$$+ \beta \log(\frac{\sum_{a=1}^{N} \exp (V_{surp}^{a}(s', u', \sigma'))}{\sum_{a=1}^{N} \exp (V_{surp}^{a}(s, u, \sigma))}) - Q(u, s; \theta)^2]$$

At thermal equilibrium, $\sum_{a=1}^{N} \exp (V_{surp}^{a}(s, u, \sigma)) = \sum_{a=1}^{N} \exp (V_{surp}^{a}(s', u', \sigma'))$,

$$= L(\theta) = \mathbb{E}_{s,s' \sim R}[\frac{1}{2}(r + \gamma \mathbb{E}_{s' \sim R}[\log \sum_{u' \in A} \exp Q(u', s'; \theta^{-})])$$

$$+ \beta \log(\frac{\sum_{a=1}^{N} \exp (V_{surp}^{a}(s', u', \sigma'))}{\sum_{a=1}^{N} \exp (V_{surp}^{a}(s', u', \sigma'))}) - Q(u, s; \theta)^2]$$

$$= L(\theta) = \mathbb{E}_{s,s' \sim R}[\frac{1}{2}(r + \gamma \mathbb{E}_{s' \sim R}[-\log \sum_{u' \in A} \exp Q(u', s'; \theta^{-})] + \beta \log(1) - Q(u, s; \theta)^2)]$$

Equation 10 represents the Soft Q-Learning objective, hence proving the result.

3 Complete Results

3.1 Energy-based Surprise Minimization

In this section we present the complete results of EMIX agents for all the 12 scenarios considered in StarCraft II micromanagement. While some scenarios depict significant performance improvements, other scenarios present incremental gains as a result of early surprise minimization during the exploration phase.
3.1.1 Comparison to MARL agents

Figure 5: Learning comparison of success rate percentages between EMIX and state-of-the-art MARL methods for all StarCraft II micromanagement scenarios. Results are averaged over 5 random seeds with each session consisting of 2 million environment interactions. EMIX significantly improves the performance of the QMIX agent on a total of 9 out of 12 scenarios. In addition, EMIX presents less deviation between its random seeds indicating consistency in collaboration across agents.
3.1.2 Comparison to SMiRL

Figure 6: Learning comparison of success rate percentages between EMIX and SMiRL-QMIX for all StarCraft II micromanagement scenarios. EMIX improves the performance of QMIX for all the considered scenarios whereas the SMiRL scheme often presents sub-optimal convergence. Moreover, direct usage of standard deviations of the state distribution leads to significant approximation errors which induce sample-inefficient behavior. This can be observed from 3s vs 4z and 3s vs 5z scenarios wherein SMiRL fails to show any learning behavior. Additionally, EMIX presents less deviation between its random seeds indicating consistency in surprise minimization.
3.2 Ablation Study

3.2.1 Energy-based Surprise Minimization

Figure 7: Comparison of success rate percentages between EMIX, TwinQMIX (EMIX without surprise minimization) and QMIX for all 12 StarCraft II micro-management scenarios. While TwinQMIX stabilizes the performance of QMIX across agents, the surprise minimization scheme of EMIX introduces robust and sample-efficient policies.
3.2.2 Temperature Parameter

Figure 8: Comparison of success rate percentages with different $\beta$ values for EMIX on all 12 StarCraft II micromanagement scenarios. $\beta = 0.01$ is a suitable value as it balances between bellman updates and the surprise minimization objective.
4 Implementation Details

4.1 Model Specifications

This section highlights model architecture for the surprise value function. At the lower level, the architecture consists of 3 independent networks called state_net, q_net and surp_net. Each of these networks consist of a single layer of 256 units with ReLU non-linearity as activations. Similar to the mixer-network, we use the ReLU non-linearity in order to provide monotonicity constraints across agent. Using a modular architecture in combination with independent networks leads to a richer extraction of joint latent transition space. Outputs from each of the networks are concatenated and are provided as input to the main_net consisting of 256 units with ReLU activations. The main_net yields a single output as the surprise value $V_{\text{surp}}(s, u, \sigma)$ which is reduced along the agent dimension by the energy operator. Alternatively, deeper versions of networks can be used in order to make the extracted embeddings increasingly expressive. However, increasing the number of layers does little in comparison to additional computational expense.

4.2 Hyperparameters

Table 3 presents hyperparameter values for EMIX. Value of $\beta$ was tuned between 0.001 and 1 in intervals of 0.001 with best performance observed at $\beta = 0.01$. A total of 2 target $Q$-functions were used as the model is found to be robust to any greater values.

| Hyperparameters                  | Values                       |
|---------------------------------|------------------------------|
| batch size                      | $b = 32$                     |
| learning rate                   | $\alpha = 0.0005$           |
| discount factor                 | $\gamma = 0.99$             |
| target update interval          | 200 episodes                 |
| gradient clipping               | 10                           |
| exploration schedule            | 1.0 to 0.01 over 50000 steps |
| mixer embedding size            | 32                           |
| agent hidden size               | 64                           |
| temperature                     | $\beta = 0.01$              |
| target $Q$-functions            | 2                            |

Table 3: Hyperparameter values for EMIX agents