

Ensemble and Auxiliary Tasks for Data-Efficient Deep Reinforcement Learning

Muhammad Rizki Maulana^[0000–0002–3457–2563] (✉) and
Wee Sun Lee^[0000–0002–0988–2500]

School of Computing, National University of Singapore, Singapore
`rizki@u.nus.edu`, `leews@comp.nus.edu.sg`

Abstract. Ensemble and auxiliary tasks are both well known to improve the performance of machine learning models when data is limited. However, the interaction between these two methods is not well studied, particularly in the context of deep reinforcement learning. In this paper, we study the effects of ensemble and auxiliary tasks when combined with the deep Q-learning algorithm. We perform a case study on ATARI games under limited data constraint. Moreover, we derive a refined bias-variance-covariance decomposition to analyze the different ways of learning ensembles and using auxiliary tasks, and use the analysis to help provide some understanding of the case study. Our code is open source and available at <https://github.com/NUS-LID/RENAULT>.

Keywords: Deep reinforcement learning · ensemble learning · multi-task learning.

1 Introduction

Ensemble learning is a powerful technique to improve the performance of machine learning models on a diverse set of problems. In reinforcement learning (RL), ensembles are mostly used to stabilize learning and reduce variability [1, 5, 17], and in few cases, to enable exploration [22, 4]. Orthogonal to the utilization of ensembles, auxiliary tasks also enjoy widespread use in RL to aid learning [13, 18, 20, 15].

The interplay between these two methods has been studied within a limited capacity in the context of simple neural networks [29] and decision trees [27]. In reinforcement learning, this interaction – to the best of our knowledge – has not been studied at all.

Our principal aim in this work is to study ensembles and auxiliary tasks in the context of deep Q-learning algorithm. Specifically, we apply ensemble learning on the well-established Rainbow agent [10, 9], and additionally augment it with auxiliary tasks. We study the problem theoretically through the use of bias-variance-covariance analysis and by performing an empirical case study on a popular reinforcement learning benchmark, ATARI games, under the constraint of low number of interactions [14]. ATARI games offer a suite of diverse problems, improving the generality of the results, and data scarcity makes the effect of

ensembles more pronounced. Moreover, under the constraint of low data, it is naturally preferable to trade-off the extra computational requirement of using an ensemble for the performance gain that it provides.

We derive a more refined analysis of the bias-variance-covariance decomposition for ensembles. The usual analysis assumes that each member of the ensemble is trained on a different dataset. Instead, we focus our analysis on a single dataset, used with multiple instantiations of a randomized learning algorithm. This is commonly how ensembles are actually used in practice; in fact, the multiple datasets that are used for training members of the ensemble are often constructed from a single dataset through the use of randomization. Additionally, we introduce some new “weak” auxiliary tasks that provide small improvements, based on model learning and learning properties of objects and events. We show how ensembles can be used for combining multiple “weak” auxiliary tasks to provide stronger improvements.

Our case study and analysis shows that,

- Independent training of ensemble members works well. Joint training of the entire ensemble reduces Q-learning error but, surprisingly did not perform as well as an independently trained ensemble.
- The new auxiliary tasks are “weakly” helpful. Combining them together using an ensemble can provide a significant performance boost. We observe reduction in variance and covariance with the use of auxiliary tasks in the ensemble.
- Despite their benefits, using all auxiliary tasks on each predictor in ensemble may result in poorer performance. Analysis indicates that this could cause higher bias and covariance due to loss of diversity.

It is interesting to note that our ensemble, despite its simplicity, achieves better performance on 13 out of 26 games compared to recent previous works. Moreover, our ensemble with auxiliary tasks achieves significantly better human mean and median normalized performance; $1.6\times$ and $1.55\times$ better than data-efficient Rainbow [9], respectively.

2 Related Works

Reinforcement Learning and auxiliary tasks. Rainbow DQN [10] combines multiple important advances in DQN [21] such as learning value distribution [3] and prioritizing experience [23] to improve performance. Other works try to augment RL by devising useful auxiliary tasks. [13] proposed auxiliary tasks in the form of reward prediction as a classification of positive, negative, or neutral reward. [13] also proposed to predict changing pixels for the downsampled image. Recently, [18] proposed the use of contrastive loss to learn better representation. Other auxiliary tasks have been explored as well, such as depth prediction [20] and terminal prediction [15]. These auxiliary tasks are less general; they require domain with 3D inputs and problem with episodic nature, respectively. Although much research has been done with auxiliary tasks in RL, to the best of our

knowledge, none of them investigated the use of auxiliary tasks in the context of ensemble RL.

Ensemble in Reinforcement Learning. Ensemble methods have been explored in RL for various purposes [1, 22, 5, 17]. [1] investigated the effect of ensemble in RL, especially pertaining to the reduction of target approximation error. In the model-based RL, [5] used ensemble to reduce modelling errors, and [17] accelerated policy learning by generating experiences through ensemble of dynamic models. In the context of policy gradients, [7] utilized ensemble value function as a critique to reduce function approximation error. [22] proposed the use of ensemble for exploration by training an ensemble based on bootstrap with random initialization and randomly sampled policy from the ensemble. [4] extended the idea and replaced the policy sampling with UCB. Finally, [19] proposed to combine ensemble bootstrap with random initialization [22], weighted Bellman backup, and UCB [4]. While they also studied the ensemble in the similar context, they did not attempt to explain the gain afforded by the ensemble, nor did they studied the effect of combining ensemble with auxiliary tasks.

3 Background

3.1 Markov Decision Process and RL

A sequential decision problem is often modeled as a Markov Decision Process (MDP). An MDP is defined with a 5-tuple $\langle \mathbb{S}, \mathbb{A}, R, T, \gamma \rangle$ where \mathbb{S} and \mathbb{A} denote the set of states and actions, R and T represent the reward and transition functions, and $\gamma \in [0, 1)$ is a discount factor of the MDP. Reinforcement Learning aims to find an optimal solution of a decision problem of unknown MDP. One of the well known model-free RL algorithms is Deep Q Learning (DQN) [21], which learns a state-action (s, a) value function $Q(s, a; \theta)$ with neural networks parameterized by θ . The Q-function is used to select action when interacting with environment; of which the experience is accumulated in the replay buffer \mathbb{D} for learning. We refer the reader to our supplementary for more details about MDP and RL.

3.2 Rainbow Agent

The Rainbow agent [10] extends the DQN by introducing various advances in reinforcement learning. It uses Double-DQN [26] to minimize the overestimation error. Instead of sampling uniformly from the replay buffer \mathbb{D} , it assigns priority to each instance based on the temporal difference (TD) error [23]. Its architecture decomposes advantage function from value function $Q(s, a) = V(s) + A(s, a)$ and learn them in an end-to-end manner [28]. Moreover, categorical value distribution is learned in place of the expected state-action value function [3]. Thus, the loss function is given as follows. We denote the scalar-valued Q-function corresponding

to the the distributional Q-function as \hat{Q} for simplicity.

$$\mathcal{L}(\theta) = \mathbb{E}[D_{\text{KL}}[g_{s,a,r,s'}||Q(s,a;\theta)]] \quad (1)$$

$$g_{s,a,r,s'} = \Phi_{Q(s',\hat{a}';\theta')}(r + \gamma\mathcal{S}) \quad (2)$$

$$\hat{a}' = \arg \max_{a'} \hat{Q}(s', a'; \theta) \quad (3)$$

where $\Phi_{Q(s',\hat{a}';\theta')}$ denotes a distributional projection [3] based on categorical atom probabilities given by $Q(s',\hat{a}';\theta')$ for support \mathcal{S} . Q returns a column vector Softmax output with $|\mathcal{S}|$ rows instead of a scalar value and \mathcal{S} is a column vector support of the categorical distribution. The scalar-valued Q function is computed by $\hat{Q}(s,a) = \mathcal{S}^T Q(s,a)$. We refer the reader to the original paper [3] for a detailed explanation.

Multi-step returns is also employed to achieve faster convergence [24]. To aid exploration, NoisyNets [6] is utilized; it works by perturbing the parameter space of the Q-function by injecting learnable Gaussian noise.

Recently, [9] proposes a set of hyperparameters for Rainbow that works well on ATARI games under 100K interactions [14].

4 Rainbow Ensemble

Several forms of ensemble agents have been proposed in the literature [1, 22, 19] with different variations of complexities. For ease of analysis, we propose to use a simple ensemble similar to Ensemble DQN [1]. The original Ensemble DQN was not combined with the recent advances in DQN such as distributional value function [3], prioritized experience replay [23], and NoisyNets [6]. Here, we describe our ensemble, that we call *REN* (Rainbow ENsemble), which combines a simple ensemble approach with modern DQN advances in Rainbow.

REN is based on the following simple ensemble estimator:

$$Q_{ens}^{(M)}(s,a) = \sum_{m=1}^M \frac{1}{M} Q_m(s,a;\theta_m) \quad (4)$$

where θ_m is the parameter for the m -th Q function. It outputs a distributional estimate of the Q-function. The scalar-valued ensemble Q-function is given by $\hat{Q}_{ens}^{(M)}(s,a) = \mathcal{S}^T Q_{ens}^{(M)}(s,a)$. We use this Q-function as our policy by taking the action that maximizes this state-action function,

$$a = \arg \max_a \hat{Q}_{ens}^{(M)}(s,a) \quad (5)$$

The ensemble distributional Q-function is used to compute the TD target $g_{s,a,r,s'}$ following equation 2, thus allowing reduction in the target approximation error (TAE) due to averaging [1]. The agent learns the estimator by optimizing the following loss,

$$\mathcal{L}(\theta_m) = \mathbb{E}_{D_m}[D_{\text{KL}}[g_{s,a,r,s'}||Q(s,a;\theta_m)]] \quad (6)$$

where m is the index of the member of the ensemble and \mathbb{D}_m is the dataset (buffer) for the m -th member. The loss is computed for each member m and optimized independently.

Rainbow uses prioritized experience replay which assigns priority to each transition. This allows for important transitions – transitions with higher error – to be sampled more frequently. Since REN consists of multiple members, we adopt the use of multiple prioritized experience replay for each member; i.e., for each member m , we have a prioritized replay \mathbb{D}_m with priority updated following the loss $\mathcal{L}(\theta_m)$. This allows each member to individually adjust their priority based on their individual errors, thus potentially enabling the ensemble to perform better.

Finally, NoisyNets is used for exploration in Rainbow. It takes the form of stochastic layers in the Q-function; each stochastic layer samples its parameters from a distribution of parameters modeled as a learnable Gaussian distribution. In our case, we have M Q-functions, with each containing the stochastic layers of NoisyNets.

5 Auxiliary Tasks for Ensemble RL

Auxiliary tasks have often been shown to improve performance in learning problems. However, the combination of auxiliary tasks and ensembles has not been extensively studied, particularly in the context of reinforcement learning.

We use the following framework, where each member of the ensemble can be trained together with a different set of auxiliary tasks, for combining ensembles with auxiliary tasks. Let $\mathcal{T}_m = \{t_{m,1}, \dots, t_{m,N_m}\}$ be the set of auxiliary tasks for member m , we seek to optimize the following loss,

$$\mathcal{L}_A(\theta_m) = \mathcal{L}(\theta_m) + \mathbb{E} \left[\sum_{n=1}^{N_m} \alpha_{m,n} \mathcal{L}_{t_{m,n}}(\theta_m) \right] \quad (7)$$

where $\alpha_{m,n}$ is the strength parameter and $\mathcal{L}_{t_{m,n}}(\theta_m)$ is the auxiliary task’s specific loss for task $t_{m,n}$. Each task may additionally include a set of parameters that will be optimized jointly with the parameters of the member.

Some questions immediately arise. Should every auxiliary task be used with every member of the ensemble? The other extreme would be using a single distinct auxiliary task with each member of the ensemble. If each auxiliary task is *weak* in the sense of only providing a small improvement, can they be combined in the ensemble to provide much stronger improvements? We examine some of these questions in our analysis and case study.

The framework can be viewed as the generalization of MTLE [29] and MTForest [27], where each member of the ensemble is trained with the auxiliary task of predicting the value of a distinct component of the input vector. An instantiation of this framework with REN as the ensemble is denoted as *RENAULT* (Rainbow ENsemble with AUxiLiary Tasks).

We propose to use model learning (i.e., learning transition function and reward function) and learning to predict properties related to objects and events as our

auxiliary tasks. Model learning has already been used in model-based RL [14], whereas predicting properties related to objects and events appears to be quite natural – rewards and hence the returns are often associated with events in the environment.

We only consider tasks that can easily be integrated with the ensemble. Some methods such as CURL [18] requires substantial changes to the base algorithm such as requiring momentum target network and data augmentation input, making their use in *RENAULT* difficult.

5.1 Network Architecture

Before delving into each auxiliary task, we will describe our network architecture in detail. Our network consists of two main components: a feature/latent state extraction function $h(s) = z$ and a latent Q function $q(z)$. The feature function h is a two layer convolution neural networks. Due to the use of dueling architecture, $q(z) = \frac{1}{|\mathbb{A}|} \sum_{a=1}^{|\mathbb{A}|} \text{adv}(z, a) + v(z)$, where $|\mathbb{A}|$ is the action space of the problem, adv is a latent advantage function, and v is a latent value function. Both adv and v are two layer fully-connected networks with ReLU as a first layer activation function. We use $\text{adv}_1(v_1)$ to denote the first layer of the $\text{adv}(v)$ function.

5.2 Model Learning as Auxiliary Tasks

Our first auxiliary tasks are based on model learning. Model learning is widely used in the context of model-based RL; but here we are using them as auxiliary tasks for DQN. They are easy to use with DQN; each task operates independently and requires no additional changes to the base algorithm. The detail on each model learning task is provided below.

Latent State Transition. We learn a deterministic latent transition function which maps a latent state $z = h(s)$ and action a to its next latent state through a parameterized function $T(z'|z, a; \theta)$. Given the actual next state s' , we seek to minimize the loss between the predicted latent state z' and $h(s')$. We use smooth L1 loss [11] as our objective function.

Inverse Dynamic. Inverse dynamic [2] is a function that learns to predict the action that causes a transition from a certain state s to another state s' . Given $z = h(s)$ and $z' = h(s')$, we seek to learn a parameterized function $T^{-1}(\hat{a}|z, z'; \theta)$ by minimizing the loss of predicted action \hat{a} with the real action a via cross entropy loss.

Reward Function. Let $w_1 = \text{adv}_1(z, \cdot)$ and $w_2 = v_1(z)$ be hidden representations corresponding to the output of the first layer of latent advantage function and latent value function for a latent state $z = h(s)$. Given an action a , and let $w = [w_1, w_2]$ be the concatenation of hidden representations w_1 and w_2 , we seek to learn a reward function $r(w, a; \theta)$ by minimizing the distributional histogram loss [12] with the real reward $r(s, a)$. The use of reward function as an auxiliary prediction is not new [13]. However, we specify the task as a distributional prediction instead of classification, generalizing their formulation.

5.3 Object and Event based Auxiliary Tasks

Our second set of auxiliary tasks aim to learn features that are useful for object and event based prediction. We propose two novel auxiliary tasks: change of moment and total change of intensity, to encourage learning features related to objects and events, respectively. The proposed tasks are simple, self-contained, and fairly general when objects and events are present, making them ideal for RENAULT. The detail of these two tasks are given as follows.

Change of moment. We adopt the concept of moment in physics; a way to account for the distribution of physical quantities based on the product of distance and the quantities. In our case, we use pixels in place of the physical quantities. Thus, the moment corresponds to the distribution of the pixels, which roughly characterizes the distribution of the objects in the screen. For a given image state $s \in \mathbb{R}^{C \times W \times H}$ with channel C , width W , and height H , the moment is computed by $\mu(s) = \frac{1}{C} \sum_c \sum_{x,y} d(x,y) \times s_{c,x,y}$ where d is a distance function to some reference point. We use coordinate $(0, 0)$ as a reference point and euclidean distance as a distance function. We learn a function that captures the change of moment between a state s and its corresponding next state s' given an action a : $\delta_\mu(z, a; \theta) \approx \mu(s') - \mu(s)$, where $z = h(s)$ is the latent state of s . For stability, we normalize the change of moment by a squared total distance given by d . The function is optimized with smooth L1 loss.

Total change of intensity. An event is often characterized by the change of total pixel intensity. For example, objects disappearing due to destruction results in the loss of total pixel intensity, spawning of enemies increases the total pixel intensity, and an explosion triggers dramatic total change of intensity. As such, learning total change of intensity can be a sufficiently strong signal to learn to associate rewards with events. Similar idea regarding learning changes of pixels intensity has been explored by [13], however, they propose to predict changes of intensity in the downsampled image patches using architecture similar to autoencoder. In contrast, we opt for a simpler objective of predicting the total change of intensity instead.

Given an image state s and its corresponding next state s' given an action a , we denote the channel-mean of the state as \hat{s} and the next state as \hat{s}' . In addition, we denote the latent state of a state s as $z = h(s)$. We seek to learn a total change of intensity function $\delta_i(z, a; \theta) \approx \|\hat{s} - \hat{s}'\|_2$. Since the value is bounded, we adopt the Histogram distributional loss similar to our reward function prediction.

6 Theoretical Analysis

In this section, we perform analysis to help understand the possible gains afforded by REN and RENAULT. We analyze the generalization error through bias-variance-covariance decomposition. Such an analysis is obviously inadequate for reinforcement learning, but we use it to potentially uncover good ways to use ensembles. We then run experiments to see which of the methods actually help for the case study.

We seek to decompose the generalization error of ensembles into bias, variance, and covariance in the form similar to one proposed by [25]. Our analysis differ in that our decomposition focuses on ensembles learned through the use of a single dataset with a randomized learning algorithm. In contrast, their decomposition assumes that each member is trained on a different dataset.

We begin with the case of a single estimator. For the purpose of analysis, we assume our targets are generated by a fixed target function corresponding to the optimal Q-function $f^*(\mathbf{x})$, possibly corrupted by noise, and that our inputs are generated by a fixed policy. We want to learn a function $f(\mathbf{x}; \theta)$ to approximate the unknown target function by using a set of N training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$. For convenience, we denote $z^N = \{z_i\}_{i=1}^N$ to be a realization of a random set $Z^N = \{Z_i\}_{i=1}^N$, where $z_i = (x_i, y_i)$ and $Z_i = (X_i, Y_i)$. The parameter θ of $f(\mathbf{x}; \theta)$ is learnt by a randomized algorithm $\mathcal{A}(r, z^N)$, where r is a random number drawn independently from a random set \mathcal{R} . We will use $f(\mathbf{x}; r, z^N)$ to refer to this parameterized function.

Given a separate test vector $Z_0 = (X_0, Y_0)$, the generalization error of the function f is $GE(f) = E_{Z^N, \mathcal{R}}[E_{Z_0}[(Y_0 - f(X_0; \mathcal{R}, Z^N))^2]]$. Let

$$\begin{aligned} \text{Var}(f|X_0) &= \mathbb{E}\left[\left(f(X_0; \mathcal{R}, Z^N) - \mathbb{E}[f(X_0; \mathcal{R}, Z^N)]\right)^2\right] \\ \text{Bias}(f|X_0) &= \mathbb{E}[f(X_0; \mathcal{R}, Z^N)] - f^*(X_0) \end{aligned}$$

where \mathbb{E} denotes the expectation $\mathbb{E}_{\mathcal{R}, Z^N}$, and let $\sigma^2 = \mathbb{E}_{X_0, Y_0}[(f^*(X_0) - Y_0)^2]$ be an irreducible error.

Theorem 1 (Generalization error of random algorithm). *The generalization error of the estimator f can be decomposed as follows.*

$$GE(f) = E_{X_0}[\text{Var}(f|X_0) + \text{Bias}(f|X_0)^2] + \sigma^2. \quad (8)$$

All proofs are provided in the supplementary material.

Now, we will consider the case of ensemble estimators. Let there be M estimators $\{f_m\}_{m=1}^M$; each estimator f_m is trained using algorithm \mathcal{A} with an individual random number $r^{(m)}$. Note that $r^{(m)}$ is a realization of $\mathcal{R}^{(m)}$. Given an input \mathbf{x} , the output of the ensemble is:

$$f_{ens}^{(M)}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^M f(\mathbf{x}; r^{(m)}, z^N) \quad (9)$$

Following equation 8, the generalization error is given by

$$GE(f_{ens}^{(M)}) = \mathbb{E}_{X_0}[\text{Var}(f_{ens}^{(M)}|X_0) + \text{Bias}(f_{ens}^{(M)}|X_0)^2] + \sigma^2. \quad (10)$$

Let

$$\begin{aligned} \text{Cov}(f_m, f_{m'}|X_0) &= \mathbb{E}\left[\left(f(X_0; \mathcal{R}^{(m)}, Z^N) - \mathbb{E}[f(X_0; \mathcal{R}^{(m)}, Z^N)]\right)\right. \\ &\quad \left.\left(f(X_0; \mathcal{R}^{(m')}, Z^N) - \mathbb{E}[f(X_0; \mathcal{R}^{(m')}, Z^N)]\right)\right]. \end{aligned}$$

$$\begin{aligned}\overline{\text{Bias}}(X_0) &= \frac{1}{M} \sum_{m=1}^M \text{Bias}(f_m|X_0) \\ \overline{\text{Var}}(X_0) &= \frac{1}{M} \sum_{m=1}^M \text{Var}(f_m|X_0) \\ \overline{\text{Cov}}(X_0) &= \frac{1}{M(M-1)} \sum_m \sum_{m' \neq m} \text{Cov}(f_m, f_{m'}|X_0).\end{aligned}$$

Theorem 2 (Generalization error of ensemble with random algorithm).

The generalization error of the ensemble estimator $f_{ens}^{(M)}$ can be decomposed as:

$$GE(f_{ens}^{(M)}) = \mathbb{E}_{X_0} \left[\overline{\text{Bias}}(X_0)^2 + \frac{1}{M} \overline{\text{Var}}(X_0) + \left(1 - \frac{1}{M}\right) \overline{\text{Cov}}(X_0) \right] + \sigma^2 \quad (11)$$

Theorem 1 and 2 follow the proof from [8] and [25], respectively. Although the results look similar, there are subtle differences in terms of the assumption with respect to the availability of multiple datasets and the use of randomness.

By analysing, the relationship between $\overline{\text{Var}}(X_0)$ and $\overline{\text{Cov}}(X_0)$, we obtain the following results about REN.

Theorem 3. $\overline{\text{Var}}(X_0) \leq \overline{\text{Cov}}(X_0)$. Hence, if ensemble estimator $f_{ens}^{(M)}$ consists of M identical estimators f that differ only in the random numbers used, then $GE(f_{ens}^{(M)}) \leq GE(f)$.

This result states that ensembles with the members trained the same way cannot hurt performance. The equal case can happen, e.g. when algorithm \mathcal{A} performs convex optimization, it will converge to the same minima regardless of random number used, resulting in all members of the ensemble being the same. In contrast, non-convex optimization algorithms such as SGD converges to a minima that depends on the randomness, thus will likely result in lower error due to reduction in the covariance term. Hence, REN will achieve *at least* the same performance as Rainbow, and possibly better, under the idealized assumptions.

Instead of training each member of the ensemble $f(\mathbf{x}; r^{(m)}, z^N)$ separately, training the entire ensemble $f_{ens}^{(M)}(\mathbf{x})$ directly on the training set would result in lower training set error and possibly better generalization. We experiment with this as well in the case study.

For a single network, auxiliary tasks usually reduce the variance as they provide additional information that help constrain the network. However, this may come at the cost of additional bias as the network needs to optimize for multiple objectives. To further understand the effects of auxiliary losses, we decompose the ensemble squared bias.

Proposition 1. The $\overline{\text{Bias}}(X_0)^2$ of ensemble estimator $f_{ens}^{(M)}$ can be decomposed as follows (X_0 is omitted for readability),

$$\frac{1}{M^2} \left[\sum_m \text{Bias}(f_m)^2 + \sum_m \sum_{m' \neq m} \text{Cob}(f_m, f_{m'}) \right] \quad (12)$$

where $\text{Cob}(f_m, f'_m|X_0) = \text{Bias}(f_m|X_0)\text{Bias}(f'_m|X_0)$ denote the product of bias; we refer to this as co-bias.

This result suggests that lower ensemble generalization error can be obtained by increasing the number of negative co-bias by having a diverse set of positively and negatively biased members. This is more likely to be achieved in RENAULT if each member is assigned a unique set of auxiliary tasks. In contrast, assigning the same set of auxiliary tasks to each member results in $\overline{\text{Bias}}(X_0) = \text{Bias}(f|X_0)$ because $\forall_m \text{Bias}(f_m|X_0) = \text{Bias}(f|X_0)$.

Proposition 2. Let $\bar{f}_{m,Z} = \mathbb{E}_{\mathcal{R}^{(m)}}[f_m|Z^N]$ be a conditional expectation of f_m over random number $\mathcal{R}^{(m)}$ conditioned on Z^N . If the estimators are trained independently, then, $\text{Cov}(f_m, f'_m) = E_{Z^N}[(\bar{f}_{m,Z^N} - E[f_m])(\bar{f}'_{m',Z^N} - E[f_{m'}])]$.

This result suggests that RENAULT may also reduce covariance if appropriate auxiliary tasks are assigned to each member. Otherwise, if all members are of the same model f , then $\text{Cov}(f_m, f'_m) = E_{Z^N}[(\bar{f}_{Z^N} - E[f])^2]$ which is the variance of the averaged estimator.

Limitations. Our analysis assumes a fixed target; this is not available in RL. Instead, we have an estimate of the target (optimal Q value) based on TD return. The dataset in RL is also generated by a non-stationary policy, thus the distribution of the dataset keeps on changing during learning. Additionally, exploration also plays an important role in the learning of Q-function. Thus, it is important to note that our analysis will only provide partial insights regarding the methods; it serves to suggest possible ways to improve the algorithms, but the suggestions may not always help.

7 Experiments

In this section, we perform a set of experiments on REN and RENAULT. We compare them to prior methods. We examine whether joint training is better than independent training. We examine whether the auxiliary tasks help the ensembles and how to best use the auxiliary tasks. Before delving into each experiment, we will explain the problem domain of our case study, our architecture and hyperparameters, and our methods in detail.

Problem Domain. We evaluate REN and RENAULT on a suite of Atari games from Atari Learning Environment (ALE) benchmark. We follow the evaluation procedure of [14]; particularly, we limit the environment interaction to 100K interactions (400K frames with action repeated 4 frames) and evaluate on a subset of 26 games. We measure the raw performance score and human-normalized score, calculated as $100 \times (\text{Method score} - \text{Random score}) / (\text{Human score} - \text{Random score})$.

Architecture and hyperparameters. We follow the data-efficient Rainbow (DE-Rainbow) architecture and hyperparameters proposed by [9] and made no change to them. REN introduces a hyperparameter M which controls the number of members of the ensemble. RENAULT introduces task and member specific hyperparameters $\alpha_{m,n}$ that control the strength of each auxiliary task.

Additionally, each auxiliary task adopts different architecture; we give their description in the supplementary.

In our preliminary experiment, we found that $M < 5$ degrades performance and higher M does not increase performance significantly while requiring more resources. Thus we fix $M = 5$ throughout the experiment.

Our Methods. REN has two variants; one that is canonical according to our description in Section 4, and one that optimizes all members jointly, which we refer to as *REN-J*. RENAULT also has two variants based on how we distribute the auxiliary tasks. The first variant, which we simply refer to as RENAULT, follows the suggestion of the preceding section to distribute the auxiliary tasks. As the number of auxiliary tasks equals to the number of members, we simply assign one *unique* task for each member. In contrast, the second variant assigns *all* tasks to each member, thus we call this variant *RENAULT-all*. For simplicity, RENAULT uses $\alpha_{m,n} = 1$ for all member m and task n . For RENAULT-all, we set $\alpha_{m,n} = \frac{1}{N_m}$, where $N_m = 5$ is the number of auxiliary tasks for member m . This is to ensure that the auxiliary tasks do not overwhelm the main task.

Further experimental details can be found in the supplementary materials.

Table 1. Performance on ATARI games on 100K interactions. Human Mean and Human Median indicate the mean and the median of the human-normalized score. The last two rows show the number of games won against DE-Rainbow and REN, respectively.

	SimPLe	OT-Rnbw	DE-Rnbw	REN	RNLT	REN-J	RNLT-all
alien	405.2	824.7	739.9	828.7	883.7	800.3	890.0
amidar	88.0	82.8	188.6	195.4	224.4	120.2	137.2
assault	369.3	351.9	431.2	608.5	651.4	504.0	524.9
asterix	1089.5	628.5	470.8	578.3	631.7	645.0	520.0
bank_heist	8.2	182.1	51.0	63.3	125.0	64.7	92.3
battle_zone	5184.4	4060.6	10124.6	17500.0	14233.3	12666.7	9000.0
boxing	9.1	2.5	0.2	10.9	5.1	5.2	4.9
breakout	12.7	9.8	1.9	3.7	3.4	2.7	3.0
chopper_command	1246.9	1033.3	861.8	713.3	896.7	980.0	563.3
crazy_climber	39827.8	21327.8	16185.3	16523.3	39460.0	23613.3	22123.3
demon_attack	169.5	711.8	508.0	759.3	693.0	665.5	822.7
freeway	20.3	25.0	27.9	28.9	29.3	24.5	29.4
frostbite	254.7	231.6	866.8	2507.7	1210.3	2284.7	1167.0
gopher	771.0	778.0	349.5	246.7	542.7	521.3	323.3
hero	1295.1	6458.8	6857.0	3817.2	6568.8	6499.3	7260.5
jamesbond	125.3	112.3	301.6	518.3	628.3	276.7	420.0
kangaroo	323.1	605.4	779.3	753.3	540.0	893.3	840.0
krull	4539.9	3277.9	2851.5	3105.1	2831.3	2667.2	3827.0
kung_fu_master	17257.2	5722.2	14346.1	12576.7	15703.3	9616.7	13423.3
ms_pacman	762.8	941.9	1204.1	1496.0	2002.7	1240.7	1705.0
pong	5.2	1.3	-19.3	-16.8	-12.0	-18.7	-10.8
private_eye	58.3	100.0	97.8	66.7	66.7	-35.2	100.0
qbert	559.8	509.3	1152.9	1428.3	583.3	2416.7	1014.2
road_runner	5169.4	2696.7	9600.0	11446.7	13280.0	5676.7	7550.0
seaquest	370.9	286.9	354.1	622.7	671.3	555.3	387.3
up_n_down	2152.6	2847.6	2877.4	3568.0	4235.7	3388.0	3459.0
Human Mean	36.45%	26.41%	28.54%	41.36%	45.64%	30.78%	38.32%
Human Median	9.85%	20.37%	16.14%	20.41%	25.08%	21.97%	23.42%
vs DE-Rnbw	10 (-3)	12 (-1)	-	20 (+7)	21 (+8)	18 (+5)	19 (+6)
vs REN	8 (-5)	10 (-3)	6 (-7)	-	17 (+4)	8 (-5)	13 (0)

Table 2. Measurement of bias approximation, variance, covariance, irreducible error σ^2 , and an approximation of generalization error ($\widehat{\text{GE}}$) of all methods. For Rainbow, $\widehat{\text{Bias}}$, $\overline{\text{Var}}$, and $\overline{\text{Cov}}$ denotes the estimator bias, variance, and covariance, respectively.

	$\widehat{\text{Bias}}^2$	$\overline{\text{Var}}$	$\overline{\text{Cov}}$	σ^2	$\widehat{\text{GE}}$
REN	0.08	1.09	0.99	1.02	2.28
RENAULT	0.08	0.82	0.66	0.63	1.58
RENAULT-all	0.09	0.81	0.71	0.74	1.72
REN-J	0.07	1.07	0.51	0.52	1.41
Rainbow	0.08	0.84	-0.70		1.81

7.1 Comparison to Prior Works

We compare the performance of REN and RENAULT to SimPLe [14], data-efficient Rainbow (DE-Rainbow) [9], and Overtrained Rainbow (OT-Rainbow) [16]. Two other recent works, CURL [18] and SUNRISE [19] use game-dependent hyperparameters instead of using the same hyperparameters for all games, making their results not directly comparable to ours. The results are given in Table 1. We report the mean of three independent runs for our methods. We take the highest reported scores for SimPLe and human baselines, as they are reported differently in prior work [26, 16].

REN improves the performance of its baseline, data-efficient Rainbow on 20 out of 26 games and achieves better performance on 13 games. It also improves the mean and median human normalized performance $1.45\times$ and $1.26\times$, respectively. RENAULT further enhances the performance of REN, gaining $1.6\times$ and $1.55\times$ mean and median human normalized performance improvements. Additionally, it won on 21 games when compared to data-efficient Rainbow and exceeds REN’s performance on 17 games.

7.2 Bias-Variance-Covariance Measurements

To gain additional insights into our methods, we perform an empirical analysis by measuring their bias, variance, and covariance. Measuring bias requires the optimal Q-function which is unknown in RL. We measure the approximation to ensemble bias $\widehat{\text{Bias}}(\theta)$ based on TD return in place of the real bias. We denote the ensemble bias based on this approximation as $\widehat{\text{Bias}}$. The detail of the measurements is given in the supplementary.

The result of the measurements is given in Table 2.

We can see from Table 2 that $\overline{\text{Cov}} < \overline{\text{Var}}$ in REN as expected from Proposition 2. If the datasets used in the ensembles had been independent as well, we would have $\overline{\text{Cov}} = 0$, so the effects of independent randomization is more limited. RENAULT reduced the variance of REN as expected from the use of auxiliary tasks and running different tasks on different members of the ensemble appears to further reduce the covariance of RENAULT.

Comparison of REN and Rainbow also shows that our bias-variance-covariance measurements are not adequate for perfectly understanding the performance of the different algorithms. In particular, the generalization error of Rainbow is smaller than REN but REN had better performance. It is possible that the bias estimate using TD return is not a good proxy for the real bias; the TD return may be arbitrarily far from the optimal Q. Another possible reason could be that RL is much more than generalization error, which does not capture other aspects of RL such as exploration.

7.3 On Independent Training of Ensemble

Jointly optimizing all members of the ensemble would give better training error and possibly better generalization error. We compare the performance of REN with its variant, REN-J, that directly optimize the following loss:

$$\mathcal{L}(\theta_{ens}) = \mathbb{E}[D_{\text{KL}}[g_{s,a,r,s'} || Q_{ens}^{(M)}(s, a; \theta_{ens})]] \quad (13)$$

where $\theta_{ens} = \{\theta_m\}_{m=1}^M$. Since REN-J is essentially one single big neural network, it uses a single prioritized experience replay \mathbb{D} which is updated based on $\mathcal{L}(\theta_{ens})$.

Table 2 shows that REN-J indeed generalized better than REN. In particular, joint optimization substantially reduced Cov. However, REN surprisingly gives better overall performance compared REN-J. REN improves upon REN-J on 18 out of 26 games. It also improves the mean human normalized performance $1.34\times$, although with a slight reduction of median performance of $0.93\times$. When compared to data-efficient Rainbow, REN gains on two more games than REN-J.

Contrary to expectation, in this case study, it is preferable to train an ensemble by optimizing each member independently, rather than treating the ensemble as a single monolithic neural network and optimize all members jointly to reduce its generalization error.

7.4 The Importance of Auxiliary Tasks

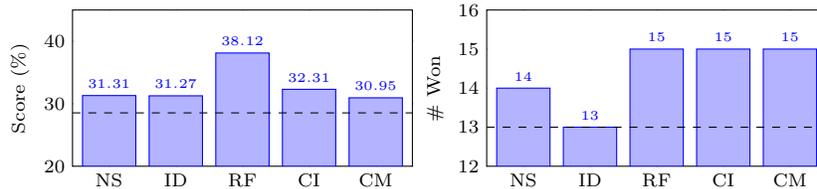


Fig. 1. Human normalized mean score (Left) and the number of games won (Right) of each member of the ensemble with (NS) latent next state prediction, (ID) inverse dynamic, (RF) reward function, (CI) total change of intensity, (CM) change of moment. As a reference, the performance of data-efficient Rainbow is indicated by a dotted line.

Table 1 shows that RENAULT improves REN on all counts. It wins on 17 games, gained $1.1\times$ and $1.23\times$ human mean and median normalized performance,

as well as increasing the win count against Rainbow to 21 games. This demonstrates the significant benefit of augmenting ensembles with auxiliary tasks, at least in this case study. Moreover, this is achieved without any tuning to the auxiliary task hyperparameter $\alpha_{m,n}$; we simply set it to 1 for all member m and task n . We also simply distribute the auxiliary tasks as such that each member is augmented with one unique task. Careful tuning of the hyperparameter and task distribution may yield even better performance improvements.

To understand the role of each auxiliary task, we analyze each of their contribution. Figure 1 shows the contribution of each member of the ensemble that is endowed with a particular auxiliary task. It is interesting to see that although each task is weakly helpful (only offers modest performance improvement), they offer significant performance boost when combined with ensembles. The best performing auxiliary tasks in terms of games won are reward function, total change of intensity (CI), and change of moment (CM) prediction. This demonstrate the usefulness of our novel auxiliary tasks; we discuss this more in the supplementary.

In the opposite extreme, inverse dynamic (ID) seems to be less useful among the auxiliary tasks. Surprisingly, retraining RENAULT without ID reduces its performance substantially (see supplementary). This suggests that ensemble improvements are not merely from individual gain, but also from diversity, through improved co-bias and covariance.

7.5 On Distributing the Auxiliary Tasks

Our theoretical result suggests that distributing the auxiliary tasks may be better than assigning all tasks on each member of the ensemble. To confirm this, we compare RENAULT its variant which assigns *all* auxiliary tasks to each member, RENAULT-all.

Table 1 shows that RENAULT-all performs worse than RENAULT, achieving lower mean and median human normalized score; this is in line with our expectation. While it may also be the case that suboptimal hyperparameters plays some roles in causing the performance degradation, this comparison is fair as we also did not perform tuning for RENAULT.

Finally, RENAULT-all has larger ensemble bias and covariance compared to RENAULT in Table 2. The larger ensemble bias could be because each network now has to optimize for more objectives. Propositions 1 and 2 also suggest that RENAULT could be benefiting from reduced co-bias and covariance. The reduction could potentially be due to each member being less correlated when trained on the same dataset compared to RENAULT-all.

8 Conclusions

In this work, we study ensembles and auxiliary tasks in the context of deep Q-learning. We proposed a simple agent that creates an ensemble of Q-functions based on Rainbow, and additionally augments it with auxiliary tasks. We provide theoretical analysis and an experimental case study. Our methods improve

significantly upon data-efficient Rainbow. We show that, although each auxiliary task only improves performance slightly, they significantly boost performance when combined using an ensemble.

Our study focuses on the interaction between ensembles, auxiliary tasks, and DQN on learning. However, RL is a multi-faceted problem with many important components including exploration. Future work includes studying their interaction with exploration, which may provide important insights and answers to some of the questions which eludes our understanding in this work.

Acknowledgements

We thank Lung Sin Kwee for useful discussions. This research is supported in part by the National Research Foundation, Singapore under its AI Singapore Program (AISG Award No: AISG2-RP-2020-016).

References

1. Ansel, O., Baram, N., Shimkin, N.: Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In: International Conference on Machine Learning. pp. 176–185. PMLR (2017)
2. Badia, A.P., Sprechmann, P., Vitvitskiy, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., et al.: Never give up: Learning directed exploration strategies. In: International Conference on Learning Representations (2019)
3. Bellemare, M.G., Dabney, W., Munos, R.: A distributional perspective on reinforcement learning. In: International Conference on Machine Learning. pp. 449–458 (2017)
4. Chen, R.Y., Sidor, S., Abbeel, P., Schulman, J.: Ucb exploration via q-ensembles. arXiv preprint arXiv:1706.01502 (2017)
5. Chua, K., Calandra, R., McAllister, R., Levine, S.: Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In: Advances in Neural Information Processing Systems. pp. 4754–4765 (2018)
6. Fortunato, M., Azar, M.G., Piot, B., Menick, J., Hessel, M., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., et al.: Noisy networks for exploration. In: International Conference on Learning Representations (2018)
7. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: International Conference on Machine Learning. pp. 1587–1596 (2018)
8. Geman, S., Bienenstock, E., Doursat, R.: Neural networks and the bias/variance dilemma. *Neural computation* **4**(1), 1–58 (1992)
9. van Hasselt, H.P., Hessel, M., Aslanides, J.: When to use parametric models in reinforcement learning? In: Advances in Neural Information Processing Systems. pp. 14322–14333 (2019)
10. Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., Silver, D.: Rainbow: Combining improvements in deep reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)

11. Huber, P.J.: Robust estimation of a location parameter. In: Breakthroughs in statistics, pp. 492–518. Springer (1992)
12. Imani, E., White, M.: Improving regression performance with distributional losses. In: International Conference on Machine Learning. pp. 2157–2166 (2018)
13. Jaderberg, M., Mnih, V., Czarnecki, W.M., Schaul, T., Leibo, J.Z., Silver, D., Kavukcuoglu, K.: Reinforcement learning with unsupervised auxiliary tasks. arXiv preprint arXiv:1611.05397 (2016)
14. Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R.H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al.: Model based reinforcement learning for atari. In: International Conference on Learning Representations (2019)
15. Kartal, B., Hernandez-Leal, P., Taylor, M.E.: Terminal prediction as an auxiliary task for deep reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. vol. 15, pp. 38–44 (2019)
16. Kielak, K.: Do recent advancements in model-based deep reinforcement learning really improve data efficiency? arXiv preprint arXiv:2003.10181 (2020)
17. Kurutach, T., Clavera, I., Duan, Y., Tamar, A., Abbeel, P.: Model-ensemble trust-region policy optimization. In: International Conference on Learning Representations (2018)
18. Laskin, M., Srinivas, A., Abbeel, P.: Curl: Contrastive unsupervised representations for reinforcement learning. In: International Conference on Machine Learning. pp. 5639–5650. PMLR (2020)
19. Lee, K., Laskin, M., Srinivas, A., Abbeel, P.: Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. arXiv preprint arXiv:2007.04938 (2020)
20. Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A.J., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., et al.: Learning to navigate in complex environments. arXiv preprint arXiv:1611.03673 (2016)
21. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. *nature* **518**(7540), 529–533 (2015)
22. Osband, I., Blundell, C., Pritzel, A., Van Roy, B.: Deep exploration via bootstrapped dqn. In: Advances in neural information processing systems. pp. 4026–4034 (2016)
23. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. arXiv preprint arXiv:1511.05952 (2015)
24. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction (2011)
25. Ueda, N., Nakano, R.: Generalization error of ensemble estimators. In: Proceedings of International Conference on Neural Networks (ICNN'96). vol. 1, pp. 90–95. IEEE (1996)
26. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 30 (2016)
27. Wang, Q., Zhang, L., Chi, M., Guo, J.: Mtforest: Ensemble decision trees based on multi-task learning. (2008)
28. Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., Freitas, N.: Dueling network architectures for deep reinforcement learning. In: International conference on machine learning. pp. 1995–2003. PMLR (2016)
29. Ye, Q., Munro, P.W.: Improving a neural network classifier ensemble with multi-task learning. In: The 2006 IEEE International Joint Conference on Neural Network Proceedings. pp. 5164–5170. IEEE (2006)