One Encoder to Rule them All: Representation Learning for Model-free Visual Reinforcement Learning using Fourier Neural Operators

Parag Dutta, Mohd Ayyoob, Shalabh Bhatnagar and Ambedkar Dukkipati
Department of Computer Science and Automation
Indian Institute of Science, Bangalore, KA, IN - 560012
{paragdutta, ayyoobmohd, shalabh, ambedkar}@iisc.ac.in

Abstract

Representation learning lies at the core of deep reinforcement learning. Although CNNs have traditionally served as the primary models for encoding image observations, modifying the encoder architecture introduces challenges, especially due to the necessity of determining a new set of hyperparameters. To address this problem, we propose a powerful representation learning technique for visual reinforcement learning utilizing Fourier Neural Operators (FNO). Our findings demonstrate that the proposed FNO encoder effectively learns representations from images that encapsulate the underlying differential equations (PDEs) governing the dynamics of the environment in an online model-free RL framework. We demonstrate the applicability of our proposed architecture by replacing the CNN image encoder in PPO, A2C, and Rainbow (a Policy Gradient, Actor-Critic, and Q-Learning RL algorithm, respectively). We achieve state-of-the-art scores (in the model-free RL setting) at both the CARLA Autonomous Driving (from image observations) benchmark and the Atari 100k benchmark. Our proposed FNO encoder is compatible with all model-free reinforcement learning algorithms, enhances both rewards and sample efficiency by implicitly learning the underlying dynamics of the environment, and eliminates the need for additional hyperparameter tuning.

1. Introduction

Despite the fact that research in contemporary Reinforcement Learning (RL) has been conducted for over fifty years, it was approximately ten years ago that RL gained significant traction when RL techniques demonstrated human-level capabilities in Atari 2600 games [22]. Since then, RL has been successfully applied to a plethora of tasks ranging from games such as AlphaGo [29] (where the RL algorithm beat the world's best GO player) to autonomous navigation [2, 9, 33]. To achieve a good general performance

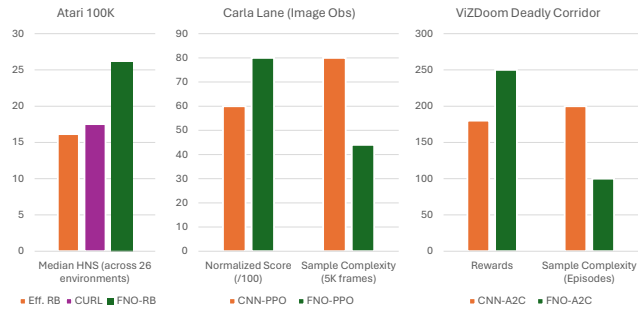

Figure 1. [Best viewed in color] **FNO's performance improvement and sample efficiency**. **Left**: The median Human Normalized Score (HNS) across 26 environments in the Atari100K benchmark. The number of frames is fixed to 100K. Efficient Rainbow (Eff. RB)[21] is the base algorithm. CURL [19] modifies Eff. RB with momentum contrast, whereas we replace the CNN encoder with our proposed FNO encoder. **Center**: The performance and sample complexity with the FNO-PPO algorithm compared to the CNN-PPO algorithm on the CARLA Lane Keeping task as defined in the D4RL benchmark, *i.e.* only images are used for control and not other information such as state, LIDAR, and overhead segmentation map. **Right**: The episodic rewards and the sample complexity of CNN-A2C vs our FNO-A2C on the ViZDoom Deadly Corridor environment.

in these complex tasks, it is necessary that learning mechanisms force agents to learn to solve the tasks from raw sensory inputs. Visual RL is one such subdomain in RL where observations in the form of images are made available to the agents, for instance, the sensor can be a camera.

Making sequential decisions under uncertainty from images is an extremely challenging problem, specifically due to the immense combinatorial possibilities within the image space (see Section 2). Naturally, a nonlinear function approximator, for instance, a CNN, is used to infer meaningful state representations from a given image. In addition to the inherent difficulties of training reinforcement agents, such as managing non-stationary targets [36], the use of function approximations for image encoding presents its own unique

challenges. Moreover, non-linear function approximations result in encoders that are (i) extremely sensitive to the input domain, and (ii) reliant on the selection of encoder architecture, weight initialization, and training hyperparameters such as learning rate, discount factor, replay buffer size, target network update frequency, etc. The first aspect warrants further discussion, which we will illustrate with examples: consider an image sensor integrated into an autonomous agent. If the training images lack diversity, the agent may struggle to perform its designated task when the sensor is upgraded to a newer model that captures images at a higher resolution, leading to an inability to adapt to the domain shift [13].

If the image rendering time is prolonged in a specific simulated environment, low-resolution images may be utilized during training to conserve wall-clock time and expedite the training process, while high-quality images can be rendered during inference. In both scenarios, the fundamental dynamics of the environment and the task the agent must complete remain unchanged. Given that convolutional neural networks (CNNs) are highly sensitive to input size, following a domain shift, the images must either be resized to their original dimensions—potentially introducing artifacts or distortions that necessitate additional interactions with the environment for adaptation — or a new encoder capable of processing the images in the updated resolution must be trained from scratch.

The aforementioned challenges can be effectively mitigated through the use of an encoder that comprehends the fundamental dynamics of the environment rather than concentrating solely on the image pixels. In our work, we tackle the problem of efficient representation learning [3] by modifying the architecture of the encoder. Although some recent works do explore the need for good representation learning [38], the choice of the encoder architecture remains surprisingly under-explored, primarily due to the newer candidate encoders being (i) difficult to train and (ii) gaining minimal advantage over the conventional CNN either in performance or sample efficiency. Thus, the following characteristics are desirable for an *ideal encoder*: (i) being independent of image dimensions, (ii) being easy to incorporate into existing algorithms, (iii) requiring minimal to no additional hyper-parameter tuning.

Our contributions:

1. Motivated by the advantages of learning the underlying dynamics of the environment from images, we propose a Fourier Neural Operator [20] based image encoder to approximate the PDEs governing the dynamics using parametric function approximators.
2. We demonstrate the broad applicability of our approach by substituting the traditional CNN encoder with our proposed FNO encoder in Proximal Policy Optimization (PPO) [27], Advantage Actor-Critic (A2C) [23], and Rainbow [6]. These algorithms are representative of model-free RL algorithms that fall under the class of policy gradient, actor-critic, and Q-learning algorithms, respectively. We establish that, without altering the hyperparameters of the base model, the integration of the FNO encoder with RL algorithms results in enhanced performance and sample efficiency compared to the standard CNN encoder.
3. We successfully achieve zero-shot domain adaptation from low-resolution training images to high-resolution inference images without necessitating any additional fine-tuning in the environment, all while maintaining performance integrity.
4. Furthermore, we conduct training in state-based environments utilizing the Soft Actor-Critic (SAC) [12] algorithm, following the replacement of the Multi Layered Perceptron (MLP) encoder with our FNO-based encoder. With this, we demonstrate the applicability of FNO encoders in state-based RL as well.

We achieve SOTA in the Atari 100k benchmark with 26.1 median HNS and 81 mean normalized score in the CARLA lane-keeping task (from the D4RL [10] benchmark). To the best of our knowledge, our work is the first major architectural breakthrough in online model-free visual RL since the invention of DQN [22].

2. Background

2.1. Reinforcement Learning

We model a sequential decision-making problem under uncertainty as a Markov Decision Process (MDP) in the infinite horizon discounted reward setting specified by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \rho, \gamma)$, where $\mathcal{S}$ is the set of possible states of an environment, $\mathcal{A}$ is the set of actions that can be taken by the agent, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta\mathcal{S}$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, ρ is the initial state distribution and $\gamma \in [0, 1)$ is the discount factor.

More often than not, the states $s \in S$ are not exposed to the agent directly. Rather, the agent has to make decisions based on an observation $o \in \mathcal{O}$ (the observation space). Given s, o is sampled according to the emission probability $\mathcal{E}$. The augmented tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{E}, r, \rho, \gamma)$ defines a Partially Observed MDP (POMDP) [4]. For instance, o represents an image, and tasks where decisions are based on image observations are categorized under Visual RL. Note that even for monochromatic `8-bit` images of size $n \times n$, the cardinality of $\mathcal{O}$ exceeds the number of atoms in our universe when $n \geq 6$.

Let $V^\pi(s)$ be the expected sum of discounted rewards obtained by an agent following policy $\pi : \mathcal{S} \to \Delta\mathcal{A}$ when starting in state s, viz., the value function under π. The goal of RL is to find a policy that maximizes V^π. A model-free RL algorithm (as opposed to a model-based one) does not

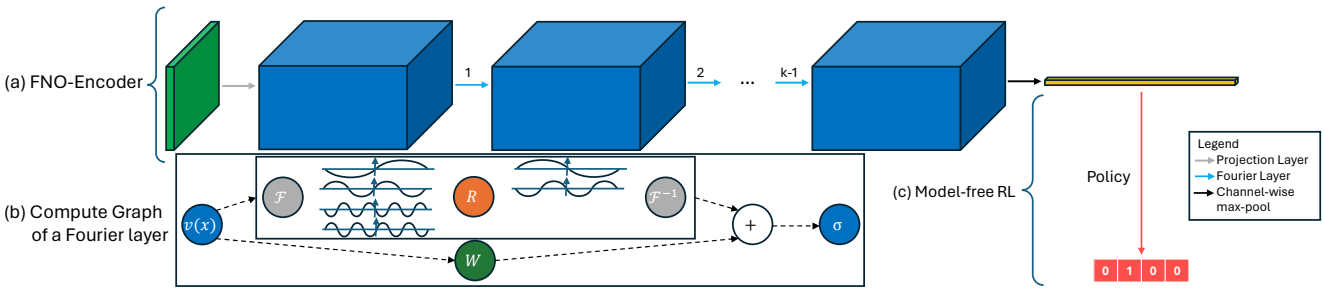

Figure 2. [Best viewed in color] **A block diagram depicting our proposed FNO encoder for model-free visual RL. (a)** We depict the image/feature maps with the cuboidal blocks and the neural network layers with the solid arrows. The image initially has c_0 channels corresponding to the history length. The observations are projected to c channels, and then $k-1$ Fourier layers are applied in succession. At each step, the height and width of the image/features remain constant. Finally, we get the representations by capturing the maximum intensities channel-wise (which makes the representations independent of the input size). **(b)** In the compute graph for the Fourier layer, the paths through which information is propagated are denoted by dashed arrows. The top path (convolution in the Fourier space) can be thought of as trying to capture the global representation since it has a low-pass filter, and the bottom path is a 1×1 convolution, which can be thought of as trying to add details to the smooth global information received from the top path. **(c)** The RL algorithm takes as input the FNO representations and outputs an action. Note that an RL algorithm may have value function(s) as well, which will take the same representations as the policy network (depicted in the diagram) as input.

estimate the $\mathcal{T}$ explicitly, but rather tries to learn the optimal policy π^* from estimates of V^π that it builds from online interactions with the environment.

The class of algorithms that directly updates π by taking small steps in the direction of the estimated gradient in the policy space is called policy gradient (PG) algorithms [30]. RL algorithms that additionally estimate the values (using Bellman backup) get categorized under actor-critic (AC) algorithms [16]. Another class of RL algorithms is Q-learning [34], which only learns the Q-value function and has an implicit policy argmax policy (See Section 3). For instance, Reinforce [35], Generalised Advantage Estimation (GAE) [26], and PPO [27] are PG algorithms, Advantage Actor-Critic (A2C), Asynchronous Advantage Actor-Critic (A3C) [23] and Soft Actor-Critic (SAC) [12] are AC algorithms, whereas Deep Q-Network (DQN) [22], C51 [25] and Rainbow [6] are Q-Learning algorithms.

2.2. Neural Operators

Neural Operators are functions that use neural network architectures to learn discretization invariant approximate mapping between infinite-dimensional function spaces [17, 24]. An important application for neural operators is learning surrogate maps for the solution operators of PDEs. Kovachki *et al.* [17] additionally show that the proposed neural operators have superior performance compared to existing machine learning-based methodologies while being several orders of magnitude faster than conventional PDE solvers.

Neural operators are formulated as iterative architectures $v_0 \mapsto v_1 \mapsto ... \mapsto v_K$ where v_j for $j = 0, 1, ..., K-1$ is a sequence of functions each taking values in $\mathbb{R}^{d_v}$. Updates to the representation $v_k \mapsto v_{k+1}$ are defined as

$$v_{k+1}(x) := \sigma\left(W v_k(x) + \left(\mathcal{K}(h;\phi)v_k\right)(x)\right), \forall x \in D, \tag{1}$$

where $\mathcal{K} : \mathcal{H} \times \Theta_\kappa \to \mathcal{L}(\mathcal{U}(D;\mathbb{R}^{d_v}))$ maps to the bounded linear operators on $\mathcal{U}(D;\mathbb{R}^{d_v})$ and is parameterised by $\phi \in \Theta_\kappa$, $W : \mathbb{R}^{d_v} \mapsto \mathbb{R}^{d_v}$ is a linear transformation, and $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function ($D \subset \mathbb{R}^d$ is a bounded, open set, $\mathcal{H} = \mathcal{H}(D;\mathbb{R}^{d_h})$ and $\mathcal{U} = \mathcal{U}(D;\mathbb{R}^{d_u})$ are separable Banach spaces [24]).

The Kernel integral operators mapping is defined as

$$\left(\mathcal{K}(a;\phi)v_k\right)(x) := \int_D \kappa_\phi\left(x, y, h(x), h(y)\right) v_k(y)\, \mathrm{d}y, \tag{2}$$

for all $x \in D$, where $\kappa_\phi : \mathbb{R}^{2(d+d_h)} \to \mathbb{R}^{2(d_v \times d_v)}$ is a neural network parameterised by ϕ. Here κ_ϕ plays the role of a kernel function which is learned from data. The kernel integral operator in Equation 2 is replaced by a convolution operator defined in the Fourier space. Let $\mathcal{F}$ denote the Fourier transform of a function $f : D \to \mathbb{R}^{d_v}$ and $\mathcal{F}^{-1}$ be its inverse.

Removing the dependence on the function h and imposing $\kappa_\phi(x, y) = \kappa_\phi(x, -y)$ in Equation 2 yields a convolution operator. After applying the convolution theorem, we get

$$\left(\mathcal{K}(a;\phi)v_k\right)(x) = \mathcal{F}^{-1}(\mathcal{F}(\kappa_\phi) \cdot \mathcal{F}(v_k))(x), \forall x \in D, \tag{3}$$

where κ_ϕ can be directly parameterized in the Fourier space.

The Fourier integral operator is defined as

$$\left(\mathcal{K}(a;\phi)v_k\right)(x) = \mathcal{F}^{-1}(R_\phi \cdot (\mathcal{F}v_k))(x), \forall x \in D, \tag{4}$$

and the resultant neural operator is termed as Fourier Neural Operator (FNO) [20].

Table 1. **Average percentage human-normalized scores (HNS%) obtained across three runs on selected games from the Atari100K benchmark**. The results for the baselines have been taken verbatim from the respective papers.

Game	Rainbow	Efficient Rainbow	CURL	FNO (Ours)
Breakout	1.74	0.69	11.1	**42.7**
Freeway	0.0	94.26	90.20	**101.01**
Pong	0.28	3.97	11.90	**36.83**
Qbert	−0.30	7.44	6.61	**18.44**
Seaquest	0.15	0.68	**0.75**	**0.75**

3. Model-free Visual Reinforcement Learning in Infinite Dimensions

In the setting of POMDPs, $\mathcal{H}$ and $\mathcal{U}$ (as defined in Section 2) are essentially the observation space $\mathcal{O}$ and action space $\mathcal{A}$, respectively. In this section, we will, by abuse of notation as well as to make our notation consistent with prior work, interchangeably use $\mathcal{O}$ and $\mathcal{S}$. Note, however, that the observation space is not the same in general as the state space since we are in the POMDP setting.

Let $\pi^* : \mathcal{O} \to \mathcal{A}$ be the optimal policy (a non-linear map), and assuming that the underlying dynamics $\mathcal{T}$ of our MDP $\mathcal{M}$ is governed by PDEs, we aim to learn an approximation of π^* by learning a parametric map $\pi_\theta : \mathcal{O} \to \mathcal{A}, \theta \in \Theta_\kappa$ for the finite-dimensional parameter space Θ_κ by choosing $\theta^\dagger$ such that $\pi(\cdot;\theta^\dagger) = \pi_{\theta^\dagger} \approx \pi^*$.

To learn in infinite dimensions, we define a cost function as $J : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$ and seek the minimizer of the problem

$$\min_{\theta \in \Theta} \mathbb{E}_s J(s, \pi(s;\theta), \pi^*(s)) \tag{5}$$

3.1. Learning with FNOs

The above setting is general enough that, to learn in infinite dimensions, we define the appropriate loss functions corresponding to the model-free RL algorithm and find the minimizer of the same.

For instance, let $Q_\theta(s,a)$ be the Q-value function, with the implicit policy $\pi_\theta(s) \equiv \arg\max_a Q_\theta(s,a)$. Assuming a trajectory buffer $B = \{s_0, a_0, r_0, s_1, ..., s_T\}$, the desired minimization problem for FNO-DQN can be formulated as

$$\min_{\theta \in \Theta_\kappa} \mathbb{E}_{s_t} \left[r_t + \gamma \max_{a_{t+1}} Q_\theta\left(s_{t+1}, a_{t+1}\right) - Q_\theta\left(s_t, a_t\right) \right]^2 \tag{6}$$

Modification in the above problem (when compared with the problem described in Mnih *et al*. [22]) for learning with FNOs instead of the conventional neural network essentially amounts to changing the parameter (search) space to Θ_κ.

Apart from being compatible with Deep RL algorithms with Q-Learning (as shown above), similar modifications can be easily applied to all other model-free RL algorithms, for instance, algorithms belonging to the class of actor-critic and policy gradient schemes. It must be noted that changing the optimization space does not change the RL algorithm but can help by learning "physics-informed" representations.

Table 2. **Average human-normalized (percentage) scores obtained across three runs on selected games from the 1% Offline RL data from the DQN Replay Dataset**. Our method modifies the R-BVE by incorporating the FNO encoder instead of the standard CNN encoder. The results for BC, QR-DQN, CQL, and DT have been taken verbatim from the Decision Transformers paper.

Game	BC	QR-DQN	CQL	DT	R-BVE	Ours
Breakout	138.9	17.1	211.1	267.5	110.5	**555.1**
Pong	85.2	18.0	111.9	106.1	92.6	**112.8**
Qbert	17.3	0.0	104.2	15.4	72.7	**110.7**
Seaquest	2.1	0.4	1.7	2.5	2.3	**3.1**

3.2. Discretization

Although the state (and subsequently observations) may exist in the continuous domain, while working with them numerically, we have to assume that we only have access to point-wise evaluations. An FNO-based policy is discretization invariant because it can learn from and evaluate functions that are discretized in an arbitrary way. Since the parameters are learned directly in the Fourier space, resolving the policy function in a physical space essentially amounts to projecting on the basis $e^{2\pi i(z,k)}$ (it is well-defined everywhere on $\mathbb{R}^d$).

4. Related Work

In the existing literature, various efforts have been made to integrate newer architectures into the practical RL algorithms. The most noteworthy of these is the Decision Transformer [7], which encodes trajectories sequential manner. However, two critical points should be highlighted: (i) a convolutional neural network (CNN) was employed to encode images into a vector, which was subsequently processed sequentially to predict actions based on rewards, and (ii) the representations were derived from offline datasets, rendering this architecture ineffective in online scenarios. However, transformer-based models have been effectively utilized to leverage long-horizon contexts in offline settings, demonstrating superior performance compared to traditional offline RL algorithms such as Quantile Regression DQN (QR-DQN) [8], Conservative Q-Learning (CQL) [18], and Regularized Behavior Value Estimation (R-BVE) [11].

The studies on representation learning that are most aligned with our approach are SAC-AE [37] and CURL [19]. SAC-AE employed a variational autoencoder to learn good representations of the input images. On the

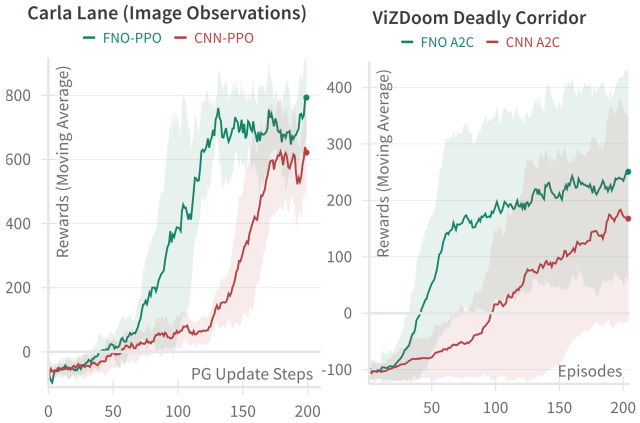

Figure 3. [Best viewed in color] **FNO results comparison with base algorithm**. **Left**: PPO is used as the base algorithm on CARLA. We show the mean and std. deviation of 3 runs. **Right**: A2C is used as the base algorithm on ViZDoom. We show the mean and standard deviation of 10 runs (more runs on this environment to accommodate high variance).

other hand, CURL utilized the MoCo [14] algorithm. This method maintains two versions of the encoder; one encoder is updated through back-propagation, while the other is updated via Polyak averaging of the modified weights.

Both techniques enhance representation learning, albeit at the expense of additional GPU memory needed to accommodate auxiliary models (a decoder in SAC-AE and an extra encoder in CURL), which are ultimately discarded after training.

On the other hand, a way to learn good representations is described in SGI [28]. However, even SGI has the requirement of the offline dataset to learn "RL-centric" representations, along with additional momentum encoders for (i) shared feature extractor training, (ii) self-predictive representation, and (iii) inverse modeling.

As mentioned earlier, the drawback of using the aforementioned works is that the methods use CNN encoders in some form. Additionally, they either use offline datasets to mitigate the instability of the online learning process or auxiliary models, which adds to the computation cost during training. On the contrary, our method is entirely online, uses no additional encoders, and requires no additional hyper-parameter tuning.

5. Experiments and Results

The implementation of our proposed FNO-based RL algorithm(s) involves replacing the traditional CNN-based encoder with an FNO-based encoder model[1] (as shown in Figure 2). Next, we evaluate the proposed FNO-based encoder against the CNN encoder in two key aspects: (i) enhancement in performance (measured by the evaluation rewards achieved), and (ii) sample efficiency (number of samples required by FNO to achieve CNNs best rewards).

[1]Codes, trained policies, and wandb [5] logs are available at: https://github.com/paragduttaiisc/FNO-RL

Table 3. **Median human-normalized (percentage) scores obtained across three runs on selected games from the Atari100K benchmark**. Instead of resizing the images to 84×84 (according to DeepMind wrapper), we use different resolutions. Since the CNNs can accept only a fixed image size, they give runtime error (we denote by: —). Our method on the other hand can accept any sized image without any degradation in performance

Image	Rainbow	Efficient Rainbow	CURL	FNO (Ours)
84×84	-0.02	16.14	17.53	**26.11**
100×100	—	—	—	**26.09**
128×128	—	—	—	**26.10**

5.1. Environments and Benchmarks

Atari

The Atari 100K benchmark introduced in [39] comprises 26 Atari 2600 games. At most 100 thousand frames, corresponding to $\sim$ 2 hours of human gameplay, are observable. The 1% Atari offline dataset consists of 500K transitions from the DQN Replay dataset from which a suitable policy must be learned in the offline setting. The action space is discrete.

ViZDoom - Deadly Corridor

This is a 3D first-person shooter (FPS) game based on the game Doom, with the image and action space similar to ALE. The task requires long-term planning by recognizing image observations [15].

CARLA - Lane

This is a 3D autonomous driving environment. We use the D4RL `carla-lane-v0` [10] setting where only image observation is present. The action space is multi-dimensional and continuous.

Deepmind Control Suite

Also known as DMC [31], comprises of 6 robotic continuous control tasks designed using the MuJoCo [32] physics engine. The observations are fully observable, and we use these set of experiments in our ablation studies. The action space is continuous and each joint of the robot needs to be controlled individually for the robot to accomplish the specified task.

5.2. Model-free (Online) RL algorithms

To demonstrate the compatibility of our FNO-based image observation encoder, we replace the vanilla CNN encoders in the following model-free RL algorithms:

a **Efficient Rainbow** [21] - We use this DQN-based algorithm on the Atari100K benchmark. For comparisons,

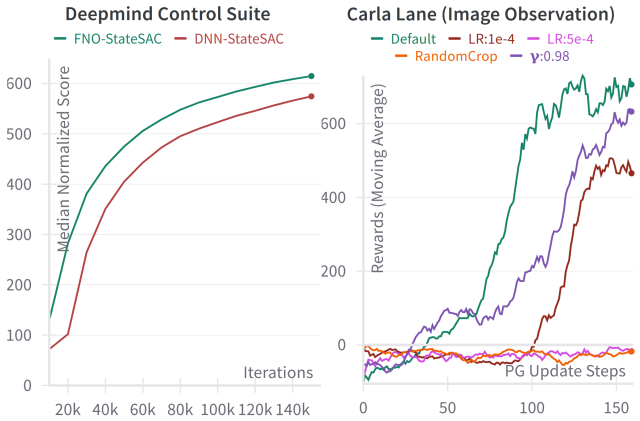

Figure 4. [Best viewed in color] **Ablation Experiments**. **Left**: We replace the DNN encoder with FNO for the StateSAC in the DMC benchmark. **Right**: Changing the hyper-parameters negatively affects the policy performance on the D4RL CARLA lane benchmark.

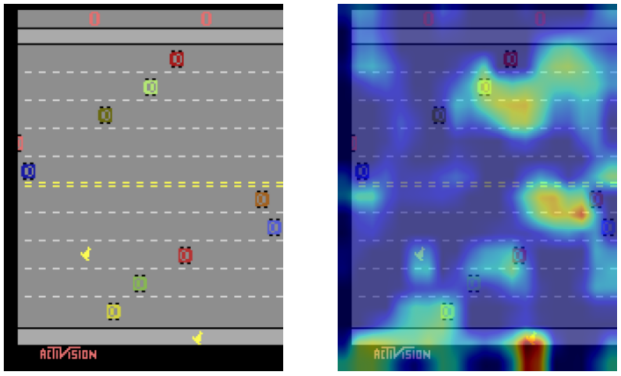

Figure 5. [Best viewed in color] **The saliency map for the Freeway environment corresponding to the optimal action**. It can be noticed that the focus is on the agent and also one lane across, although there is no car present in the focused region. If there was a car present at that location, it would have collided with the agent upon taking the ↑ action by the time the agent would have reached the concerned lane.

we use the following standard baselines, (i) Rainbow [6], (ii) Efficient Rainbow, (iii) CURL [19]

b **Advantage Actor Critic (A2C)** [23] - The synchronous version of A3C actor-critic algorithm was used on the ViZDoom environment.

c **Proximal Policy Optimization** [27] - This policy gradient-based algorithm is used in the CARLA environment.

We selected these as base algorithms because (i) they perform effectively within their specific environments, (ii) the performance of these algorithms has been independently reproduced, and (iii) the authors have provided access to their implementations. Related works such as CURL (as mentioned in Section 4) also use (a) to demonstrate their representation learning strategy in the Atari100k benchmark.

5.3. Offline Model-free RL algorithms

Similar to the online setting, we additionally run multiple experiments in the offline setting with the 1% DQN Replay dataset [1]. We use the following strong baselines to evaluate the advantage of our approach: (i) Behavior Cloning (BC), (ii) Quantile Regression Deep Q-Learning (QR-DQN), (iii) Conservative Q-Learning (CQL) (discrete version), (iv) Decision Transformers (DT), and (v) Regularized Behavior Value Estimation (R-BVE). We modify the CNN encoder in the R-BVE by replacing with our FNO encoder model and run the experiments without any change to the hyper-parameters. We followed the same experimental setting that was described in DT to report their results on the 4 Atari games. Since we are using their results verbatim, we also report our results on the aforementioned 4 Atari games.

5.4. Architecture and other details

For architecture selection, we performed behavior cloning on 2 million transitions (9^{th} and 10^{th} replay buffer split) from the Atari Offline dataset on Breakout, Pong, Freeway and MsPacman environments. We use $k = 3$ Fourier layers in our implementation. An affine layer is used for the initial projection, and a channel-wise max pooling is used for the final layer. To reduce GPU memory footprint, we down-sampled using a 3×3 max-pool layer after every Fourier layer, with a stride of 2 (we noticed no change in results with or without the down-sampling). The number of modes was fixed to 5, and the width of the channels was set to $c = 128$. GeLU activations were used throughout the encoder. (See Figure 2 for a pictorial representation). The official code for R-BVE is not available, so we implemented it from scratch. Instead of the mean squared error loss (during SARSA) described in the paper, we use the SmoothL1 loss since we find that it converges faster. In our experiments, we use the default hyper-parameters provided in the respective official code repositories. All images are preprocessed according to [22] with the exception of CARLA, where the image dimension is 48×48. We run all experiments $k = 3$ times with the seeds $k \in 0, 1, ..., k-1$ with the exception of ViZDoom where we set $k = 10$ since the variance across games was very high.

5.5. Results

5.5.1. Atari 100K

HNS Results on selected games are tabulated in Table 1 and raw rewards results for all games are reported in Table 4). Our FNO encoder outperforms the default CNN encoder in Efficient Rainbow in $20/26$ games and improves Efficient

Table 4. **Average raw scores obtained across three runs on all games from the Atari100K benchmark**. The results for the baselines have been taken verbatim from the respective sources. RB denotes Rainbow algorithm [6], and Eff. RB the Efficient Rainbow variant [21].

Game	Random	RB	Eff. RB	CURL + Eff. RB	FNO + Eff. RB (Ours)	Human
Alien	227.8	318.7	739.9	558.2	**784.7**	7127.7
Amidar	5.8	32.5	**188.6**	142.1	149.6	1719.5
Assault	222.4	231.0	431.2	600.6	**727.9**	742.0
Asterix	210.0	243.6	470.8	734.5	**903.3**	8503.3
BankHeist	14.2	15.5	51.0	131.6	**241.3**	753.1
BattleZone	2360.0	2360.0	10124.6	**14870.0**	12900.0	37187.5
Boxing	0.1	−24.8	0.2	1.2	**5.0**	12.1
Breakout	1.7	1.2	1.9	4.9	**14.0**	30.5
ChopperCommand	811.0	120.0	861.8	**1058.5**	916.7	7387.8
CrazyClimber	10780.5	2254.5	16158.3	12146.5	**21296.7**	35829.4
DemonAttack	152.1	163.6	508.0	**817.6**	745.5	1971.0
Freeway	0.0	0.0	27.9	26.7	**29.9**	29.6
Frostbite	65.2	60.2	866.8	**1181.3**	783.0	4334.7
Gopher	257.6	431.2	349.5	**669.3**	589.3	2412.5
Hero	1027.0	487.0	**6857.0**	6279.3	6099.7	30826.4
Jamesbond	29.0	47.4	301.6	**471.0**	431.7	302.8
Kangaroo	52.0	0.0	779.3	872.5	**1040.0**	3035.0
Krull	1598.0	1468.0	2851.5	**4229.6**	3172.0	2665.5
KungFuMaster	258.5	0.0	**14346.1**	14307.8	1966.7	22736.3
MsPacman	307.3	67.0	1204.1	**1465.5**	1362.0	6951.6
Pong	−20.7	−20.6	−19.3	−16.5	**−7.7**	14.6
PrivateEye	24.9	0.0	**1204.1**	218.4	75.4	69571.3
Qbert	163.9	123.5	1152.9	1042.4	**2615.0**	13455.0
RoadRunner	11.5	1588.5	**9600.0**	5661.0	4200.0	7845.0
Seaquest	68.4	131.7	354.1	**384.5**	384.0	42054.7
UpNDown	533.4	504.6	2877.4	2955.2	**2983.3**	11693.2
Median HNS	0.00	−0.02	16.14	17.53	**26.11**	100.00

Rainbow 1.36× and 6.04× on median and mean HNS, respectively. When comparing with model-free RL baselines, we achieve superior performance in 13 tasks compared to the 9 of CURL. We additionally achieve superhuman performance in the following games: Freeway, Jamesbond, and Krull, on top of achieving 26.1 median Human-Normalized Score when compared to 16.1 and 17.5 of Efficient Rainbow and CURL, respectively. Note that comparison of median HNS is a standard metric [19, 21, 37, 39].

5.5.2. Atari Offline

Results on selected games are tabulated in Table 2. As can be seen, although R-BVE performs worse than CQL and DT in the offline setting, a simple modification to the R-BVE with the FNO encoder (Ours) outperforms them in every task and by a wide margin (2.89× when compared to DT)

5.5.3. ViZDoom Deadly Corridor

The improvement in the sample efficiency and rewards obtained is clear from Figure 3. The maximum evaluation reward obtained is ~ 900.

5.5.4. CARLA Lane

Figure 3 shows the performance and sample efficiency enhancement in the CARLA lane task. Even without additional lidar/segmentation information, during the best eval runs, the agent obtained a perfect normalized score of 100 ± 0.03 (corresponding to an expert policy).

6. Ablations and Discussions

6.1. FNO encoders on fully observable MDPs

To study the generality of FNO, we apply SAC [12] on the state-based DeepMind (continuous) Control Suite benchmark [31] across six environments and plot the median normalized scores in Figure 4 (history length $c_0 = 3$). The improvement in score and sample efficiency is significant.

6.2. Effect of Hyper-parameters

In Figure 4, we additionally illustrate the rewards with adjusted hyperparameters, including learning rates, the γ value, and random cropping, a widely utilized representation learning method in supervised learning (also employed

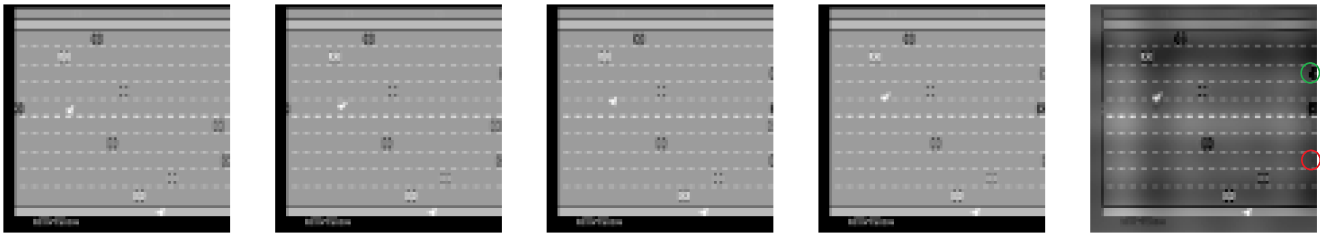

Figure 6. **Feature learning and reconstruction**. **Left 4** images are the input to the policy. Starting from left, the frames correspond to $t-3, t-2, t-1, t^{th}$ timestep. **Right** is our reconstructed image corresponding to action $\downarrow$. We find that the reconstructed frame corresponds to the next frame of the environment had the aforementioned action been taken. The details are obscure and only noticeable at the edges. The green and red circles in the reconstructed frame indicate the locations where a car enters and leaves the frame, respectively. Not only is the FNO preserving the transition information of the agent, but also all the relevant details about all the cars, *i.e*. information about the environment dynamics, is captured within the model with surprising details.

in CURL). Each of these modifications negatively impacted performance in comparison to the default parameters.

6.3. Effect of Data Augmentations

The observation of the failure of random crops suggests that the representations acquired from FNO differ from those obtained through CNNs. To delve deeper into this matter, we calculate the saliency map (see Figure 5). The policy appears to concentrate on the appropriate locations. Nevertheless, to interpret the perspective of the policy, we reconstruct and visualize the frames it perceives. We incorporate a linear regressor at the conclusion of the trained FNO encoder and distill the policy into this framework. Subsequently, we assign weights to all the penultimate feature maps based on the weights determined by the regressor. The resulting image is displayed in Figure 6. Our findings indicate that the policy implicitly forecasts the subsequent frame to facilitate suitable actions.

6.4. Zero-shot Domain Adaptation with High-Resolution Image Observations

Finally, while the standard image input size is 84×84, when the trained FNO policy was presented with images sized 100×100, and 128×128, there was no variation in the rewards or scores obtained. This indicates zero-shot generalizability and insensitivity to image resolution, as illustrated in Table 3. We observed consistent $\sim$ 26% HNS across the 26 Atari environments. This effect was even more remarkable in the Carla Autonomous Driving (lane keeping) task, where the images produced during training were only 48×48. However, during evaluation, we were able to increase the size to 224×224, and the model maintained a performance level of ~ 0.8, comparable to that of the expert policy, despite the smaller input size. It is important to note that due to the discretization characteristics of FNOs discussed in Section 3, retraining the models was unnecessary, thanks to the emergent properties of Fourier Neural Operators.

7. Limitations and Future Work

Despite conducting several experiments to determine the appropriate method for combining the channels of a specific RGB image, we were unsuccessful in achieving this goal. Consequently, FNO encoders are presently limited to functioning solely with grayscale images.

Given that FNOs were originally designed for the purpose of learning the fundamental PDEs [20] and thus are naturally suitable for forecasting subsequent states, a logical progression of our approach would involve its application in model-based reinforcement learning, particularly in the domains of Robotics and system identification. Furthermore, alternative variants of neural operators may also be investigated for reinforcement learning.

8. Conclusion

We present a Fourier Neural Operator encoder designed for model-free visual reinforcement learning, which can be effortlessly integrated as a substitute for conventional CNNs. We showcase the efficacy of the proposed encoder across various visual environments in both online and offline contexts, utilizing multiple families of RL algorithms. Furthermore, we aim to comprehend the representations acquired and the generalizability of these learned representations through comprehensive ablation experiments. FNO encoders possess a unique characteristic of being invariant to the resolution of image inputs, thereby enabling zero-shot domain adaptation.

Acknowledgments

The authors would like to thank the SERB, Department of Science and Technology, Government of India, for the generous funding through the IMPRINT Project: IMP/2019/000383.

References

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 104–114. PMLR, 2020. 6

[2] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020. 1

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. 2

[4] Dimitri P Bertsekas. *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 2nd edition, 1995. 2

[5] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com. 5

[6] Johan Samir Obando Ceron and Pablo Samuel Castro. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *International Conference on Machine Learning*, pages 1373–1383. PMLR, 2021. 2, 3, 6, 7

[7] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, pages 15084–15097. Curran Associates, Inc., 2021. 4

[8] Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI Press, 2018. 4

[9] Ambedkar Dukkipati, Rajarshi Banerjee, Ranga Shaarad Ayyagari, and Dhaval P. Udaybhai. Learning skills to navigate without a master: A sequential multi-policy reinforcement learning algorithm. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022. 1

[10] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning, 2020. 2, 5

[11] Caglar Gulcehre, Sergio Gómez Colmenarejo, Ziyu Wang, Jakub Sygnowski, Thomas Paine, Konrad Zolna, Yutian Chen, Matthew Hoffman, Razvan Pascanu, and Nando de Freitas. Regularized behavior value estimation. In *arXiv Pre-print*. arXiv:2103.09575, 2021. 4

[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018. 2, 3, 7

[13] Meera Hahn, Devendra Singh Chaplot, Shubham Tulsiani, Mustafa Mukadam, James M Rehg, and Abhinav Gupta. No RL, no simulation: Learning to navigate without navigating. *Advances in Neural Information Processing Systems*, 34:26661–26673, 2021. 2

[14] K He, H Fan, Y Wu, S Xie, and R Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 5

[15] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning, 2016. 5

[16] Vijay R Konda and John N Tsitsiklis. On actor-critic algorithms. *SIAM journal on Control and Optimization*, 42(4), 2003. 3

[17] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023. 3

[18] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1179–1191. Curran Associates, Inc., 2020. 4

[19] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020. 1, 4, 6, 7

[20] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. 2, 3, 8

[21] Alexander Long, Alan Blair, and Herke van Hoof. Fast and data efficient reinforcement learning from pixels via non-parametric value approximation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7):7620–7627, 2022. 1, 5, 7

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 1, 2, 3, 4, 6

[23] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine learning*, pages 1928–1937. PMLR, 2016. 2, 3, 6

[24] Nicholas H. Nelsen and Andrew M. Stuart. The random feature model for input-output maps between banach spaces. *SIAM Journal on Scientific Computing*, 43(5): A3212–A3243, 2021. 3

[25] Mark Rowland, Marc Bellemare, Will Dabney, Rémi Munos, and Yee Whye Teh. An analysis of categorical distributional reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 29–37. PMLR, 2018. 3

[26] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016. 3

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2, 3, 6

[28] Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron Courville. Pretraining representations for data-efficient reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021. 5

[29] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587), 2016. 1

[30] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 3

[31] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite, 2018. 5, 7

[32] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. 5

[33] Chao Wang, Jian Wang, Yuan Shen, and Xudong Zhang. Autonomous navigation of uavs in large-scale complex environments: A deep reinforcement learning approach. *IEEE Transactions on Vehicular Technology*, 68(3):2124–2136, 2019. 1

[34] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992. 3

[35] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. 3

[36] Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst continual structured non-stationarity. In *International Conference on Machine Learning*, pages 11393–11403. PMLR, 2021. 1

[37] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35 (12):10674–10681, 2021. 4, 7

[38] Fuzhen Zhuang, Zhiqiang Zhang, Mingda Qian, Chuan Shi, Xing Xie, and Qing He. Representation learning via dual-autoencoder for recommendation. *Neural Networks*, 90:83–89, 2017. 2

[39] Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020. 5, 7