1 Implementation Details for Pixel-based Experiments

Hyperparameter	Value
Augmentation	Random Translate
Observation size	$(100, 100)$
Augmented size	$(108, 108)$
Replay buffer size	100000
Initial steps	10000
Training environment steps	$1.5e6$ pendulum swingup $2.5e6$ others
Batch size	128
Stacked frames	2 pendulum swingup 3 others
Action repeat	2 walker run, hopper hop 4 others
Camera id	2 quadruped, walk 0 others
Evaluation episode length	10
Hidden units (MLP)	1024
Number of layers (MLP)	2
Optimizer	Adam
$(\beta_1, \beta_2) \rightarrow (f_{CNN}, \pi_\psi, Q_\phi)$	$(.9, .999)$
$(\beta_1, \beta_2) \rightarrow (\alpha)$	$(.5, .999)$
Learning Rate (π_ψ, Q_ϕ)	$2e-4$
Learning Rate (f_{CNN})	$1e-3$
Learning Rate (α)	$1e-4$
Critic target update frequency	2
Critic EMA τ	0.01
Encoder EMA τ	0.05
Convolutional layers	4
Number of CNN filters	32
Latent dimension	64
Non-linearity	ReLU
Discount γ	0.99
Initial Temperature	0.1

2 Implementation Details for State-based Motivation Experiments

Hyperparameter	Value
Replay buffer size	2000000
Initial steps	5000
Batch size	1024
Stacked frames	4 Flare, Stack SAC; 1 otherwise
Action repeat	1
Evaluation episode length	10
Hidden units	1024
Number of layers	2
Optimizer	Adam
$(\beta_1, \beta_2) \rightarrow (f_{CNN}, \pi_\psi, Q_\phi)$	$(.9, .999)$
$(\beta_1, \beta_2) \rightarrow (\alpha)$	$(.9, .999)$
Learning Rate (π_ψ, Q_ϕ)	$1e-4$
Learning Rate (α)	$1e-4$
Critic target update frequency	2
Critic EMA τ	0.01
Non-linearity	ReLU
Discount γ	0.99
Initial Temperature	0.1

3 FLARE vs RAD CNN Activation

As shown in Figure 1, FLARE activations focus more on parts of the agent that are in charge of controlling balance and speed, such as its legs.

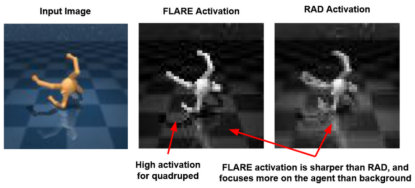

Figure 1: FLARE vs RAD CNN feature activation map.

4 Interpreting FLARE from a Two-stream Prospective

Let f_{CNN} and o'_t be the pixel encoder and the augmented observation in Flare. Then, $z_t = f_{\text{CNN}}(o'_t)$ denotes the latent encoding for a frame at time t. By computing the latent flow $\delta_t = z_t - z_{t-1}$, Flare essentially approximates $\frac{\partial z_t}{\partial t}$ via backward finite difference. Then following chain rule, we have

$$\frac{\partial z}{\partial t} = \frac{\partial z}{\partial o'} \cdot \frac{\partial o'}{\partial t} = \frac{f_{\text{CNN}}(o')}{\partial o'} \cdot \text{dense optical flow}$$

indicating that Flare eventually uses dense optical flow by propagating it through the derivative of the trained encoder. While the two-stream architecture trains a spatial stream CNN from RGB channels and a temporal stream from optical flow separately, Flare can be interpreted as training one spatial stream encoder from the RL objective and approximate the temporal stream encoder with its derivative.

5 Compare Flare with DQN Variants on Atari

Table 1: Evaluation on 8 benchmark Atari games at 100M training steps over 5 seeds. † directly taken from DQN Zoo repository and the rest are collected from our experiments.

	FLARE	RAINBOW	RAINBOW†
ASSAULT	12724±1107	**15229**±2061	13194±4348
BREAKOUT	345±22	280±18	366±20
BERZERK	**2049**±421	1636±598	3151±537
DEFENDER	**86982**±29214	44694±3984	52419±4481
MONTEZUMA	**1668**±1055	900±807	80±160
SEAQUEST	13901±8085	**24090**±12474	5838±2450
PHOENIX	60974±18044	16992±3295	**82234**±33388
TUTANKHAM	**248**±20	**247**±11	214±24

	PRIORITIZED†	QR DQN†	IQN†
ASSAULT	10163±1558	10215±1255	11903±1251
BREAKOUT	359±33	450±29	**492**±86
BERZERK	986±98	896±98	946±48
DEFENDER	13750±4182	32320±8997	33767±3643
MONTEZUMA	0±0	0±0	0±0
SEAQUEST	7436±1790	14864±3625	16866±4539
PHOENIX	10667±3142	41866±3673	35586±3510
TUTANKHAM	168±22	171±30	216±34

Table 1 compares Flare with Rainbow, Prioritized Experience Replay, QR DQN and IQN.