Deep Reinforcement Learning: Generalization

Henriette Becker Kristiansen (s190192), Sebastian Larsson (s190619)

DTU, Technical University of Denmark

Introduction

Generalisation, i.e. the ability to improve skills rather than memorising specific tasks, is a big challenge [1] in deep reinforcement learning. By using the Proximal Policy Optimisation algorithm (PPO) [4], we implement a Deep Reinforcement Learning model that can learn to play StarPilot from the Procgen [1] arcade game environment and tests how the model performs on levels it has never seen before.

Key points

- We construct a model based on the PPO algorithm found in [4], with the policy network architecture found in [3]. We then test the performance of this encoder against the Impala CNN encoder found in [2]. We test how the model generalise when trained for 8M timesteps on respectively 10, 50 and 200 levels.
- We use the ProcGen Benchmark, as described in [1], to test generalisation of our model on the game StarPilot on easy difficulty with backgrounds disabled. All testing have been done using the free compute power at Google Colab.
- Like in [4], where they introduce clipping on the objective function, we introduce clipping on the value function loss. We test with a value clipping of 0.5, and can see that for both architectures, it improves the performance considerably.
- We implement a reward modification we call *death penalty*. On every episode end a fixed number is subtracted from the reward to help encourage the notion of self preservation in the model. Our experiments show a big behavioural difference between models trained with and without death penalty implemented. Models with death penalty implemented seems to be performing better under our computational restrictions.

PPO algorithm

The Proximal Policy Optimization (PPO) algorithm is different from the Normal Policy Gradients algorithm in the following way:

- The PPO algorithm uses a clipping method on the objective to make sure that a training step does not lead to an excessively large policy update. See equations figure 1 and 2.
- PPO alternates between sampling from the environment for T timesteps, independently of actual episode lengths, and optimising the objective function.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\Big[\min(\underbrace{r_t(\theta)\hat{A}_t}_{\text{Normal Policy Gradients objective}}, \overbrace{\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t}^{\text{Clipped version of Normal Policy Gradients objective}})\Big]$$

Figure 1: Clipped PPO objective used in PPO algorithm. See [4]

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\big[\underbrace{L_t^{CLIP}(\theta)}_{\text{Clipped PPO objective}} - \underbrace{\overset{\text{Hyperparameters}}{c_1} L_t^{VF}(\theta)}_{\text{Value function loss}} + \overset{\text{Hyperparameters}}{c_2}\underbrace{S[\pi_\theta](s_t)}_{\text{Entropy term}}\big]$$

Figure 2: Final objective function used in PPO algorithm. See [4]

Encoders

Two different encoders have been considered:

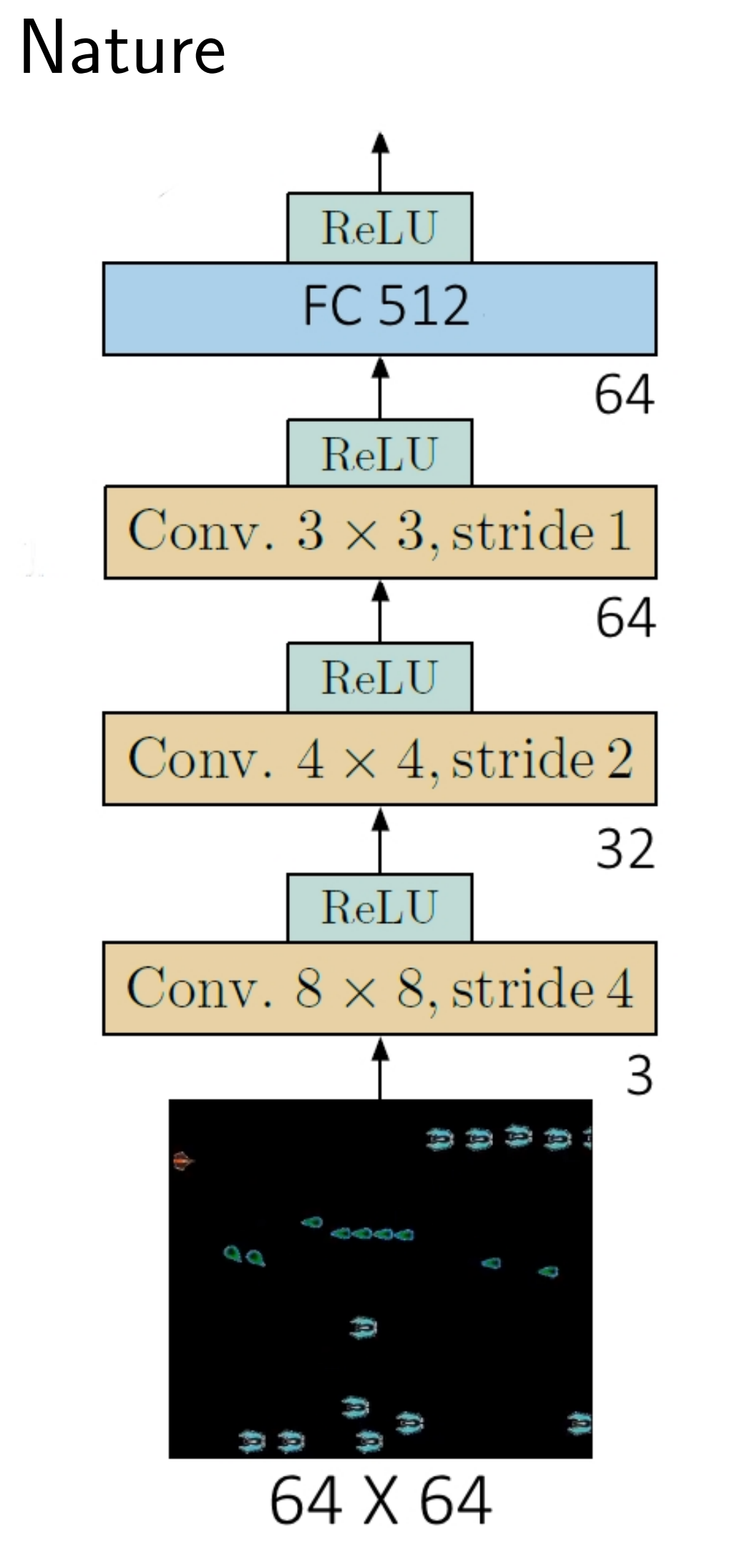

Figure 3: Nature encoder architecture. 3 hidden convolutional layers.

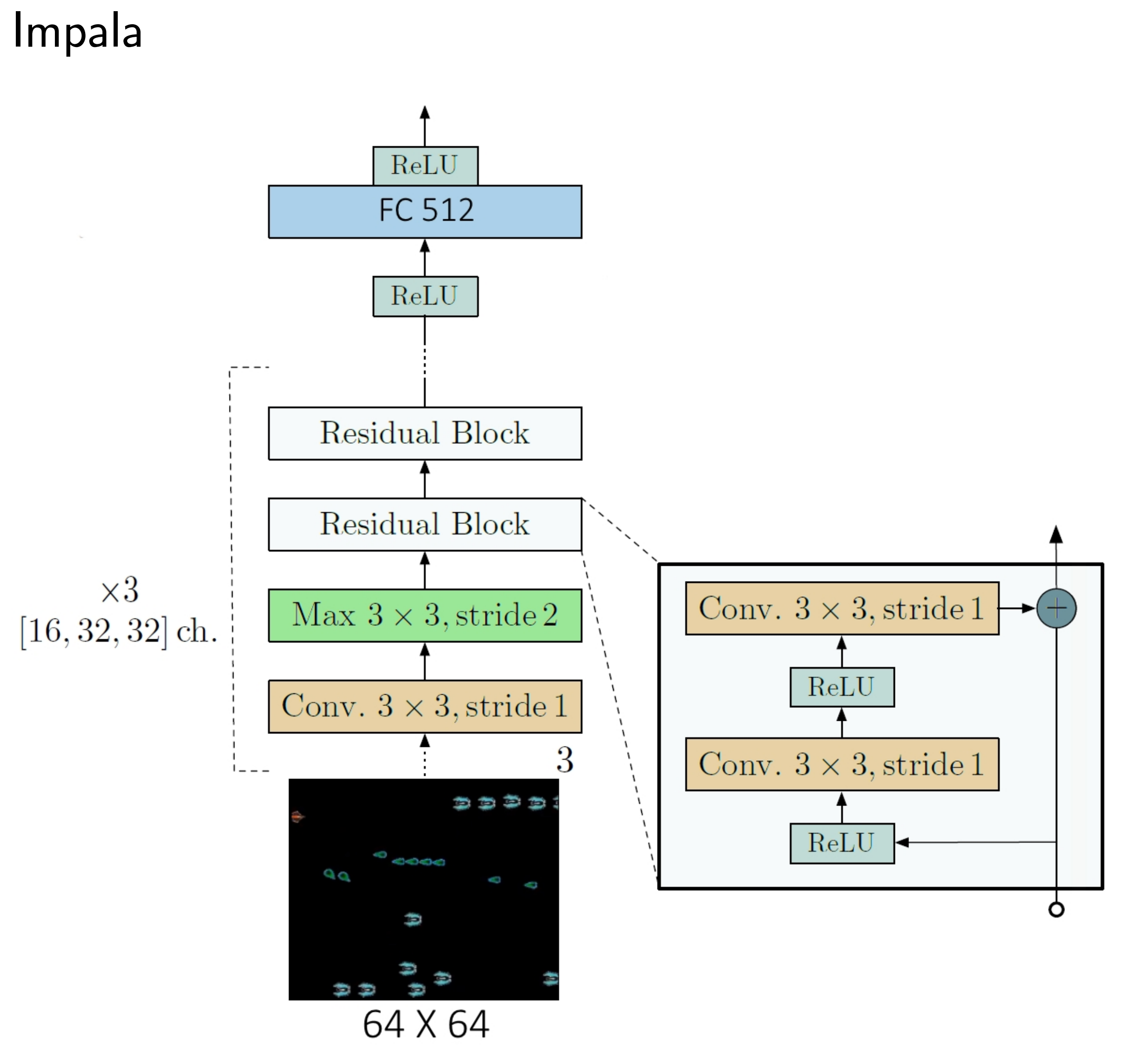

Figure 4: Impala encoder architecture. 18 hidden layers, 15 convolutional and 3 max poolings.

How well do humans play?

We have no intention of claiming how well the general world population plays this game, but based on the full population of the authors households this is how well humans play:

	Lab rat 1	Lab rat 2	Lab rat 3
Avg Score (20 levels)	14.15	14.55	18.1

Table 1: Human average over 20 levels.

Training setup

The hyperparameters seen in table 2 are inspired by [4] and [2]. It is assumed that these hyperparameters have already been optimised, so we chose not to optimise them further.

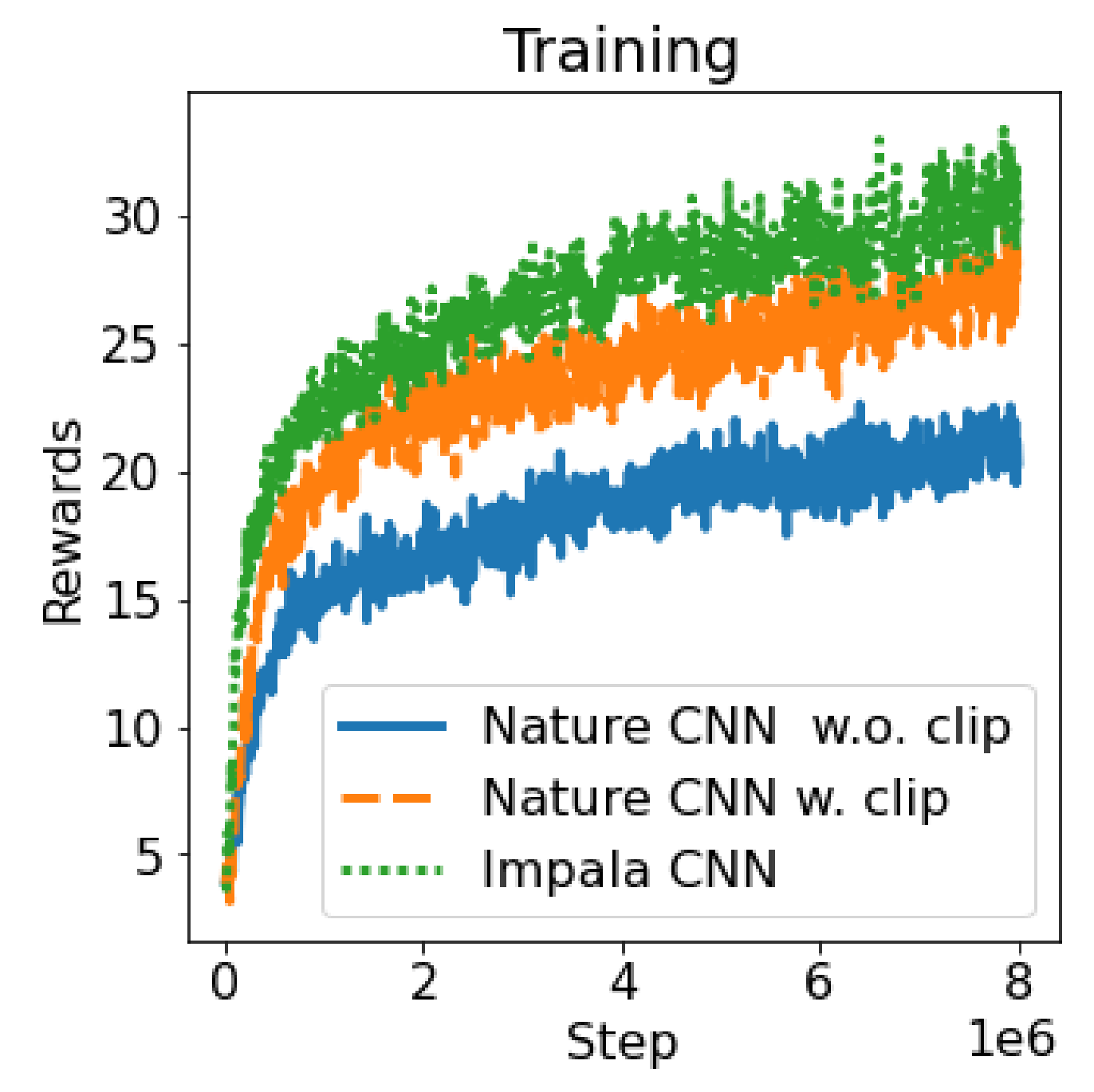

Figure 5: The model trains continually on 256 steps. When the current level ends, the next levels is started immediately.

Hyperparameter	Value
Discount γ	0.99
GAE parameter λ	0.95
# of timesteps pr rollout	256
Batch size (PPO and modified Impala)	512
Batch size (Original Impala)	32
Entropy bonus	0.01
PPO clip range	0.2
Reward Normalization	Yes
# of Workers	1
# of environments per worker	32
Total timesteps	8M
LSTM	No
Frame Stack	No
Optimizer	Adam
Learning rate (α) (Nature)	0.00025
Learning rate (α) (Impala)	0.0006

Table 2: The choice of hyperparameters for the models

Test performance

When the model is evaluated, it receives rewards based on one game, not 256 steps as in training. Trained on 10 levels for 8M timesteps we see (figure 6) that the Impala encoder outperforms the Nature encoder. We also see that that a death penalty of 1 makes no difference, but both 3, 5 and 7 seems to improve performance.

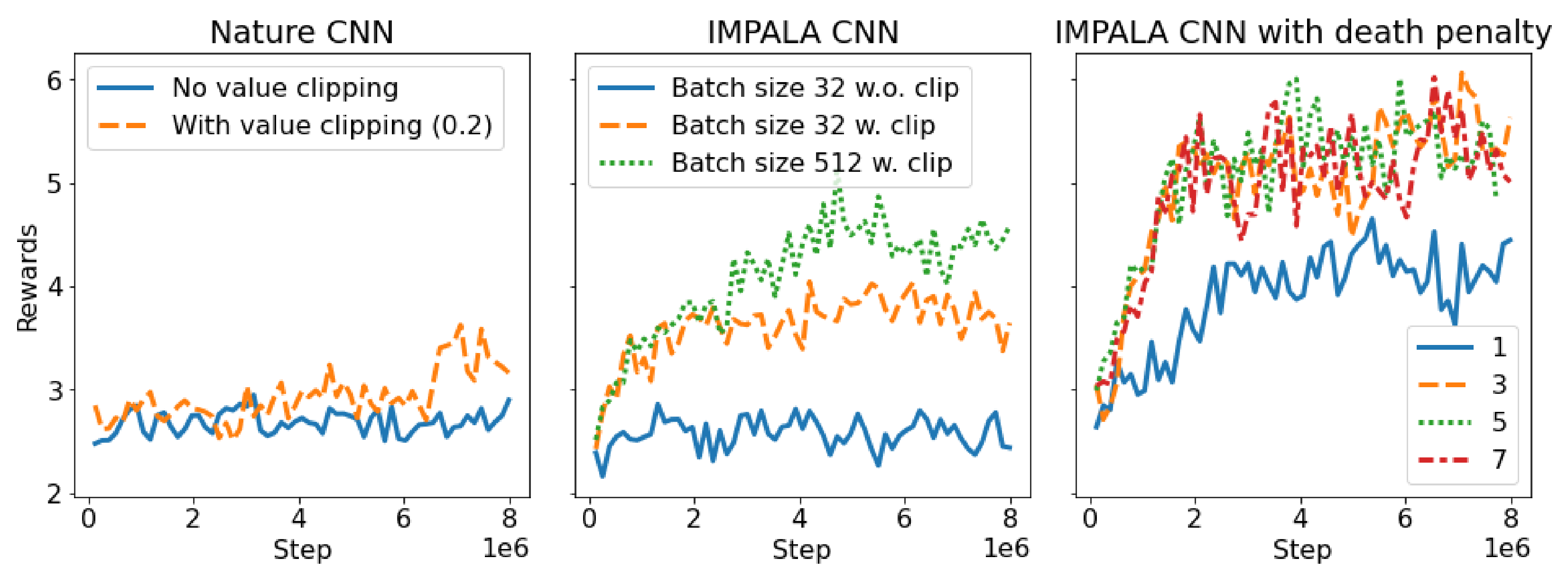

Figure 6: The left plot shows how well the Nature encoder performs in the test environment, the middle plot shows the Impala encoder's performance, and the right plot shows the test performance of the Impala encoder with different death penalties.

The number of levels that the model trains on, impacts how well it generalises, with more levels giving a better performance. As seen on figure 7, this is the case for all our models.

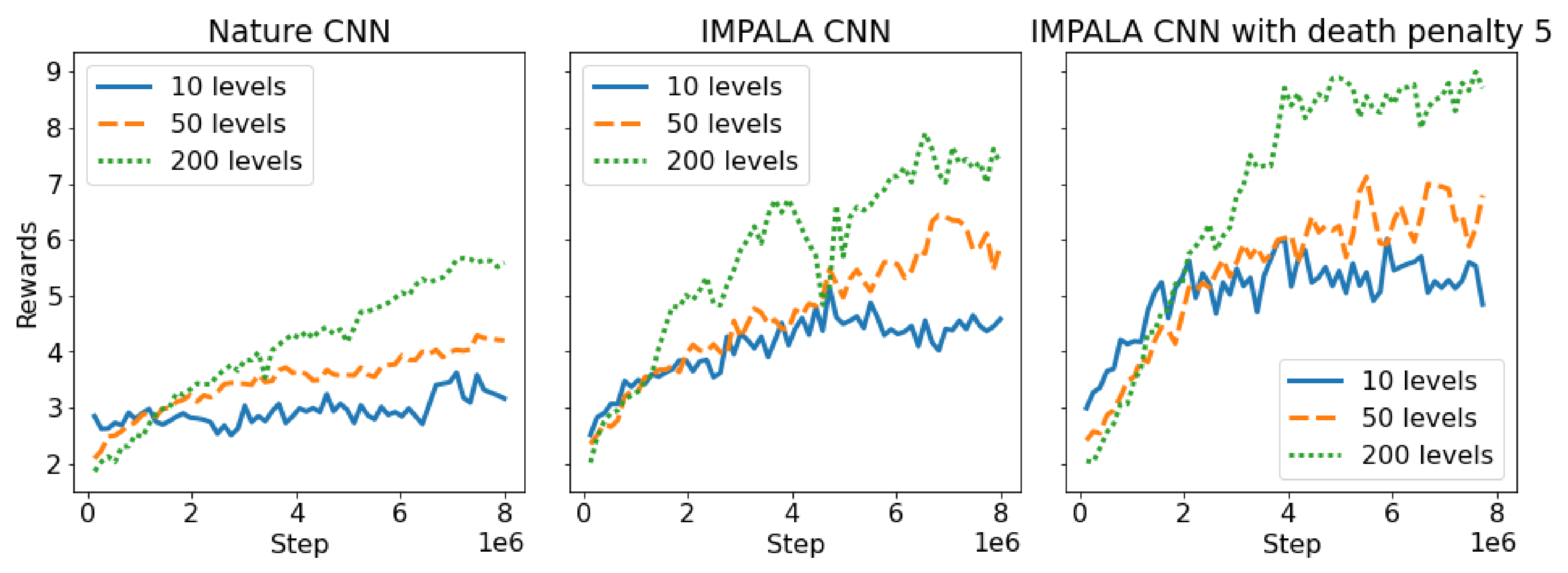

Figure 7: The plots show the test performance of the best Nature encoder, the best Impala encoder without death penalty, and the best Impala encoder with death penalty 5, all trained on 10, 50 and 200 levels.

How well do the models play?

To compare with the human baseline we evaluated the models that had trained on 200 levels for 8M timesteps on 20 new levels. The result can be seen below. Comparing table 1 with table 3 we see that the trained model performs better than our human baseline.

	Nature CNN	IMPALA CNN w.o. dp	IMPALA CNN w. dp 5
Avg Score (20 levels)	22.14	27.16	34

Table 3: Model average over 20 levels, trained on 200 levels.

References

[1] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman. Leveraging procedural generation to benchmark reinforcement learning. *CoRR*, abs/1912.01588, 2019. URL `http://arxiv.org/abs/1912.01588`.

[2] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. *CoRR*, abs/1802.01561, 2018. URL `http://arxiv.org/abs/1802.01561`.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015. ISSN 00280836. URL `http://dx.doi.org/10.1038/nature14236`.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL `http://arxiv.org/abs/1707.06347`.