Forward-Forward Optimization of Deep Q-Network

Andrii Krutsylo
Institute of Computer Science, Polish Academy of Sciences

Abstract

Deep Q-Networks (DQNs) are almost always trained with backpropagation, which builds a global computation graph and backpropagates a scalar loss through all layers. This requires storing activations, tightly couples parameters, and can be unstable for long sequences. The Forward-Forward (FF) algorithm replaces backprop with local, layer-wise objectives that only rely on forward passes and a goodness measure. We propose FF-DQN, an application of FF to value-based reinforcement learning. FF-DQN achieves comparable performance to a standard backprop, provided that the model is well-initialized to allow stable FF training, while avoiding a backward pass and reducing memory usage.

Expanding FF to RL: The representation is trained to distinguish high-value from low-value states based on current TD targets, rather than relying on artificially generated or noisy positive/negative samples.

Motivation

Forward-Forward as a memory-efficient alternative to backpropagation.

- Training is performed with forward passes only, using a local goodness objective.
- No explicit backward graph or gradient propagation through depth.
- Memory footprint is reduced because activations do not need to be stored for backprop.

Original Unsupervised Forward-Forward Algorithm

The Forward-Forward algorithm trains each hidden layer to assign high goodness to "positive" inputs and low goodness to "negative" (artificially distorted) inputs.

Goodness and loss. For a hidden activation vector $\mathbf{h}$, the goodness is

$$g(\mathbf{h}) = \sum_i h_i^2.$$

For each layer, we compute the loss

$$\mathcal{L} = \text{softplus}(m - g^+) + \text{softplus}(g^- - m),$$

where g^+ and g^- are the average goodness of positive and negative inputs respectively, and m is a margin.

Layer-wise training.

- Train one layer at a time on positive/negative pairs until it separates them according to the margin.
- Once a layer achieves sufficient separation, freeze its weights.
- Move to the next layer and repeat.

In the original formulation, this greedy layer-wise procedure avoids interference between layers at the cost of multiple training stages.

From FF to FF-DQN

We adapt the Forward-Forward (FF) principle to the Deep Q-Network (DQN) framework by training layer-wise goodness functions on real environment states. Each FF layer is optimized with a local objective, while a separate Q-head is trained concurrently using standard temporal-difference (TD) targets.

Key adaptations:

1. **Positive and negative inputs.** For each minibatch of transitions (s_t, a_t, r_t, s_{t+1}), compute TD targets
$$y_t = r_t + \gamma \max_{a'} Q_{\text{target}}(s_{t+1}, a').$$
Split the batch into two subsets:
$$\mathcal{S}^+ = \{s_t \mid y_t > \text{median}(y)\}, \quad \mathcal{S}^- = \{s_t \mid y_t \leq \text{median}(y)\}.$$
States in $\mathcal{S}^+$ are treated as positive samples, and those in $\mathcal{S}^-$ as negative samples for all FF layers.
2. **Layer-wise local update.** Each FF layer l receives batches of positive and negative inputs and minimizes the same layer-wise objective as defined in the original Forward-Forward algorithm, using the previously introduced goodness measure $g(h)$ and threshold m. Gradients are computed locally within each layer and applied via an independent optimizer.
3. **Joint training with TD head.** The FF layers and the Q-value head are trained concurrently. In each update step, the FF layers receive their local updates and the Q-head parameters are optimized with the standard TD loss
$$\mathcal{L}_{\text{TD}} = \left(r_t + \gamma \max_a Q_{\text{target}}(s_{t+1}, a) - Q(s_t, a_t)\right)^2.$$

This formulation maintains local layer objectives while integrating with online TD learning. Positive and negative states are determined by relative target values and all components are optimized in parallel during interaction with the environment.

Experimental Setup

We evaluate FF-DQN on the classic control environment CartPole-v1 from the `gymnasium`.

Environment. CartPole-v1 with episodic returns capped at 200.

Hyperparameters.

Parameter	Value
Batch size	128
Hidden dimensions	[128, 64]
Epochs	70
Buffer size	50000
Features learning rate	0.0001
Head learning rate	0.001
FF threshold m	2.0 (default for FF)

The FF model uses the same architecture as the baseline DQN, differing only in the training procedure for the hidden layers.

Results

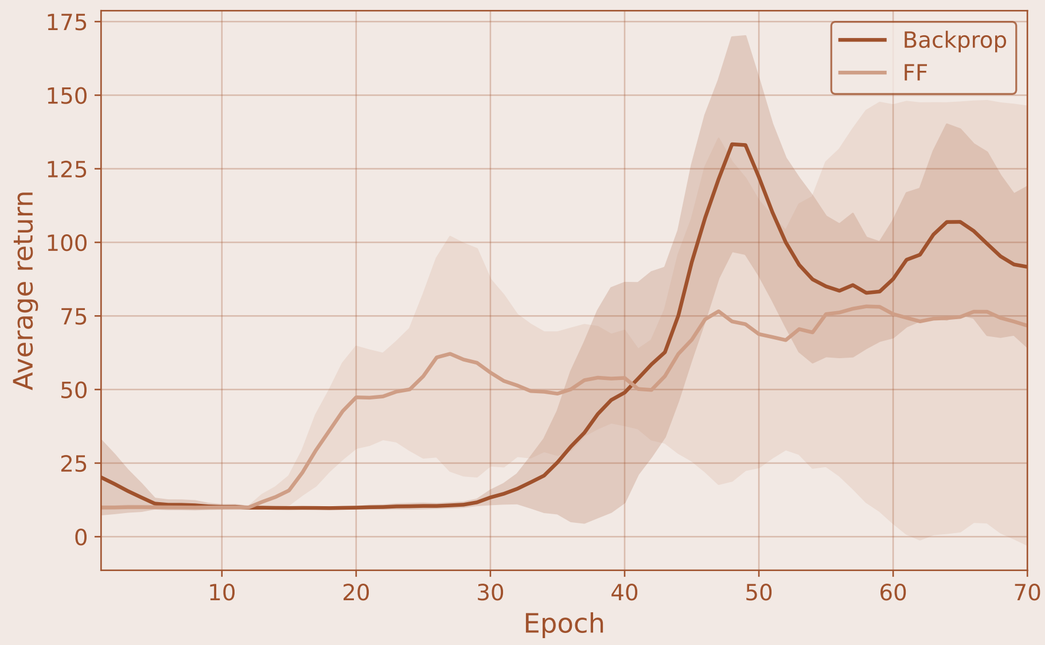

Figure 1: Average episode return during training for both agents.

Performance.

- Both methods reach near-optimal performance.
- The FF-DQN converges more slowly and exhibits higher variance across epochs, indicating reduced training stability.
- Despite slower convergence, FF-DQN ultimately achieves comparable performance to backprop-based DQN.

Discussion and Outlook

Advantages of FF-DQN.

- No explicit backward graph through the deep network, reducing memory usage.
- Each layer is optimized independently, allowing clearer inspection of learned feature representations.
- Forward-only formulation is compatible with neuromorphic or biologically inspired hardware implementations.

Limitations.

- Training is less stable, with higher variance in returns compared to backprop.
- The feature extractor is trained in an unsupervised, layer-wise manner, which may misalign with the Q-value head's objective.
- *FF-DQN is more sensitive to model initialization because early TD targets determine the positive and negative samples. If the initial model is poor, these targets are noisy, causing the updates to reinforce incorrect representations and hinder recovery.*

References

- Hinton, G. E. (2022). Forward-Forward Algorithm: Some Preliminary Investigations.

Resources

Source code (FF-DQN)

Author website

Contact: andrii.krutsylo@gmail.com | https://krutsylo.neocities.org