Sparse Representations in Deep RL: Encoding Information in Deep Architectures

Hunter Hobbs

Anuj Pasricha

Matthew Resnick

Problem

Unfortunately it hasn't changed:

Catastrophic Interference

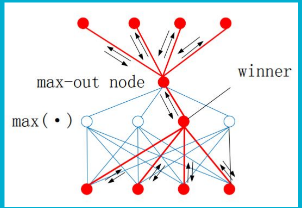
- A neural network containing a representation of a task can have significant overwriting of parameters when seeing new experiences.
- Could be a result of training on different regions of an environment for a single task or training on different tasks entirely.
- Five approaches to addressing this problem: regularization, ensembling, rehearsal, dual-memory, and sparse-coding (Kemker et. Al., 2017).
    - Our approaches are primarily sparse-coding.
    - A bit of regularization/ensembling here and there to make it happen.

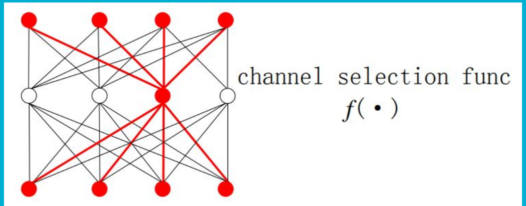
Theoretical Approach 1: Sparse Pathway Encoding

Idea: Train a deep controller normally, but encode input information onto the network structure itself.

We can do this with activation functions provided the power of trainable parameters:

- Maxout (Bengio et. Al., 2013)
- Channel-out (Wang & JaJa, 2013)





The learned network is not sparse, but a sparse representation exists as a subnetwork, accessible via activation.

Theoretical Approach 2: Distributional Regularizers

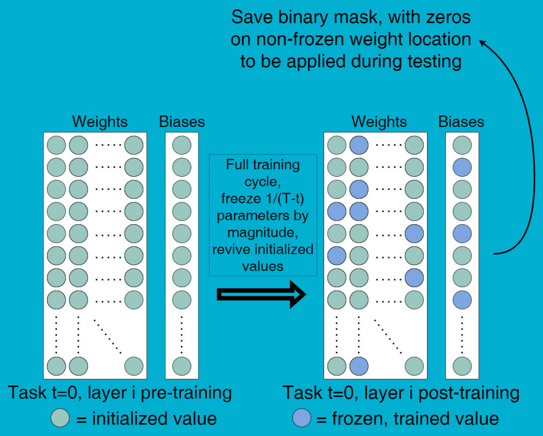
First, **KL-divergence**: a measure of difference in information between two probability distributions. Think Shannon entropy.

Then, assume the output of each hidden unit is a random variable with some distribution *p*. We can use KL-divergence as a penalty term on the loss function to encourage *p* towards another distribution with a higher density about zero. Hence, a sparse representation!

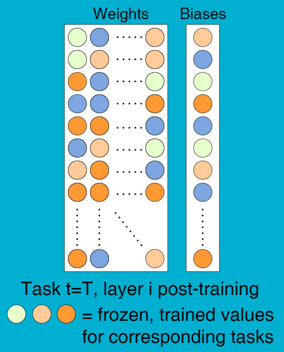
With a clipped KL-divergence, we can specify a minimum level of sparsity (Liu, 2019).

Theoretical Approach 3: Spatially Packed Task Representations

Idea: We can prune many parameters from the network and still have a good representation of a task. Instead, we should just use the remaining space for more representations of more tasks. Same network size as before pruning, but we've learned multiple sparse representations.



For T total tasks, and task number t, we train normally, then freeze 1/(T-t) tasks from training, and save a binary mask with 1s at these locations. Reinitialize, and repeat.



This is a modified PackNet, from Mallya et. Al. an algorithm for filters in a CNN for Computer Vision tasks.

All approaches are modular, so can be applied to any Deep RL algorithm.

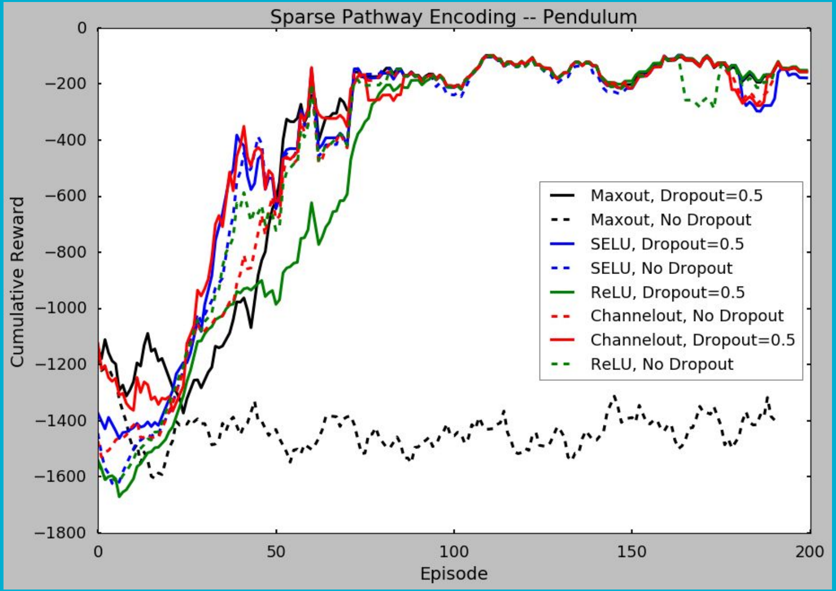
The following results were obtained from applying the approaches to the actor network of a vanilla DDPG.

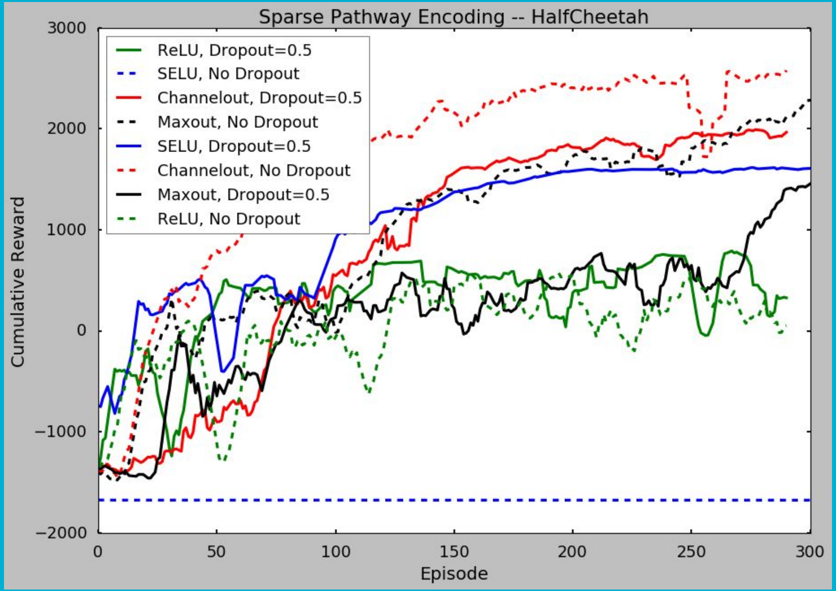
Info on other learning algorithms in the conclusion.

SPE Results

Without PER

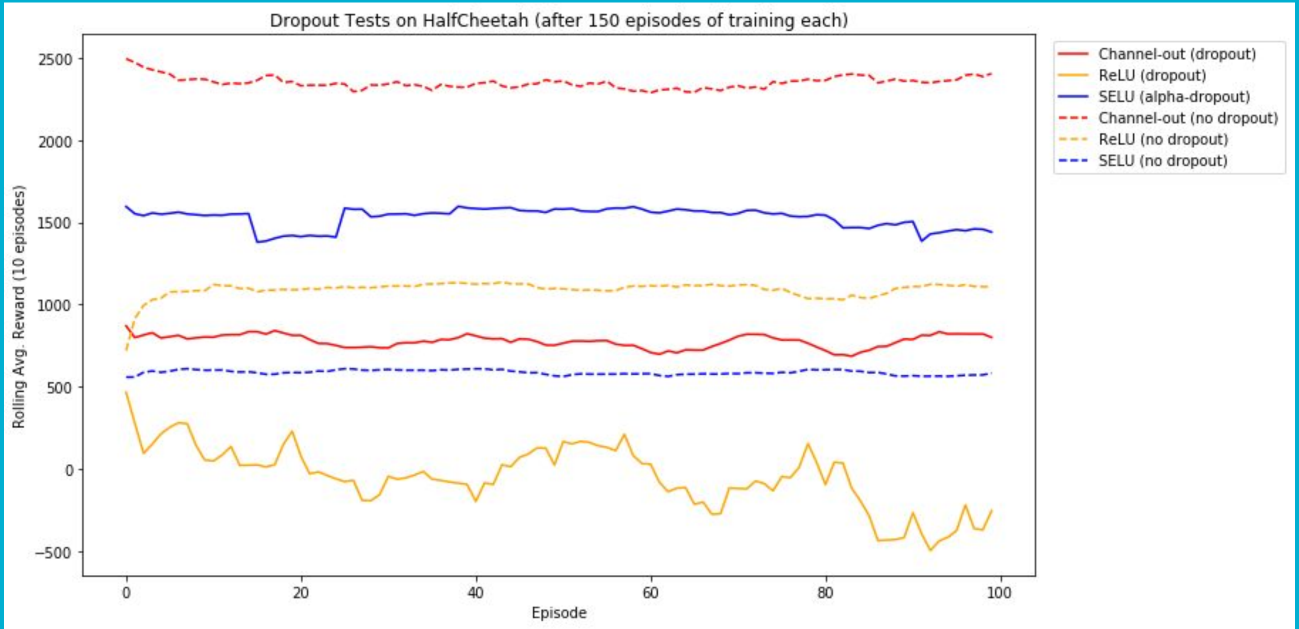
Training dynamics:



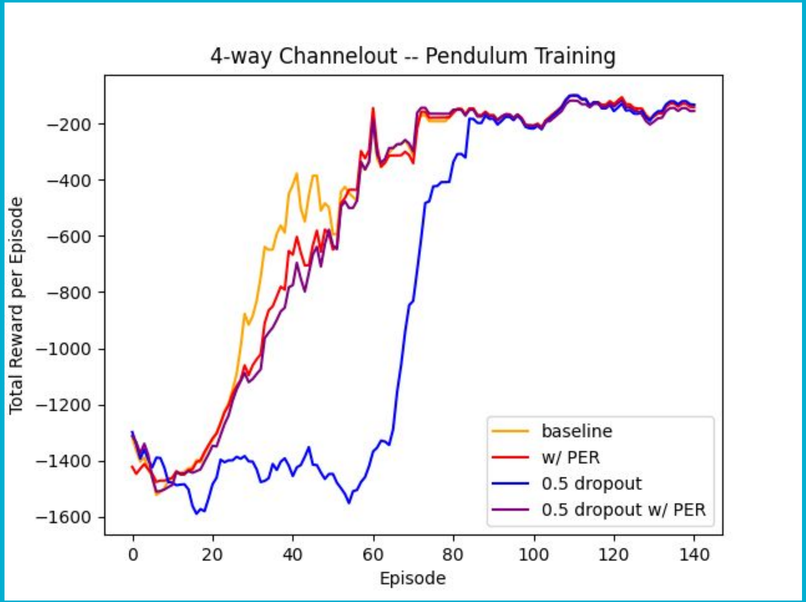


Without PER

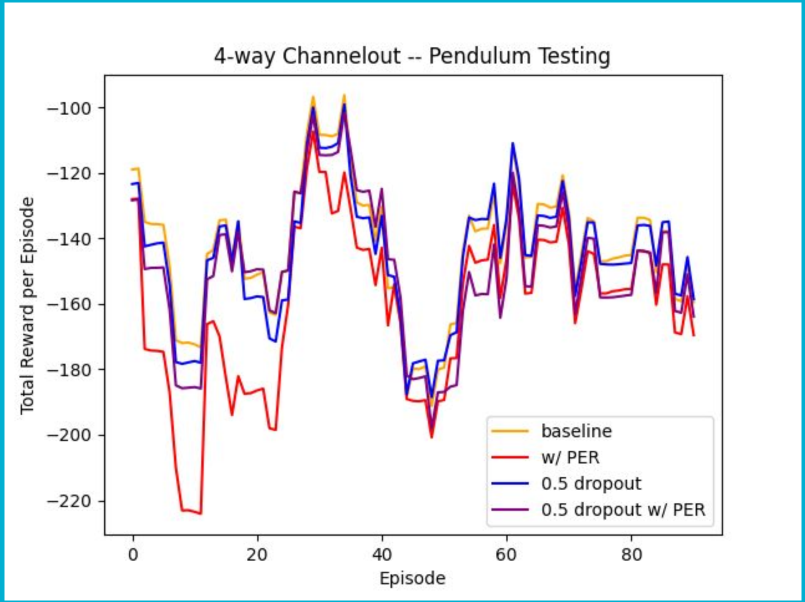
Testing:



With PER



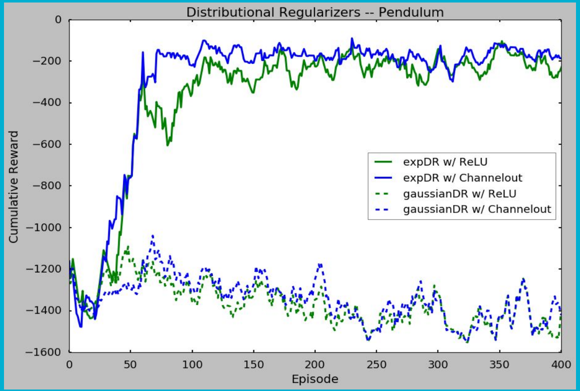
4-way Channelout -- Pendulum Training
Total Reward per Episode
Episode
−200
−400
−600
−800
−1000
−1200
−1400
−1600
0
20
40
60
80
100
120
140
baseline
w/ PER
0.5 dropout
0.5 dropout w/ PER

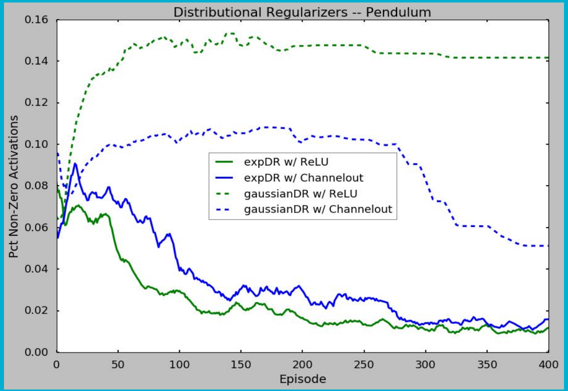


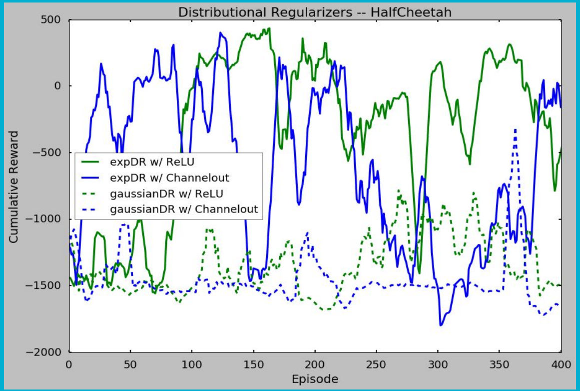
4-way Channelout -- Pendulum Testing
Total Reward per Episode
Episode
−100
−120
−140
−160
−180
−200
−220
0
20
40
60
80
baseline
w/ PER
0.5 dropout
0.5 dropout w/ PER

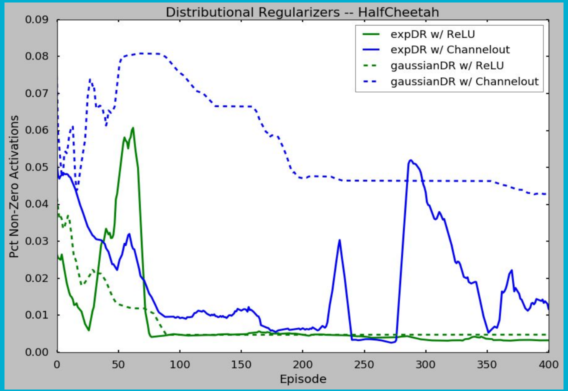
Distributional Regularizers Results

Without PER

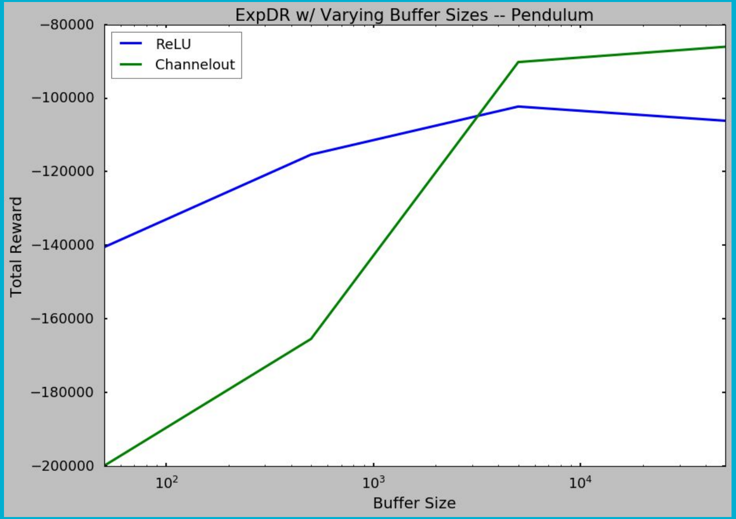

Distributional Regularizers -- Pendulum
Cumulative Reward
Episode
expDR w/ ReLU
expDR w/ Channelout
gaussianDR w/ ReLU
gaussianDR w/ Channelout

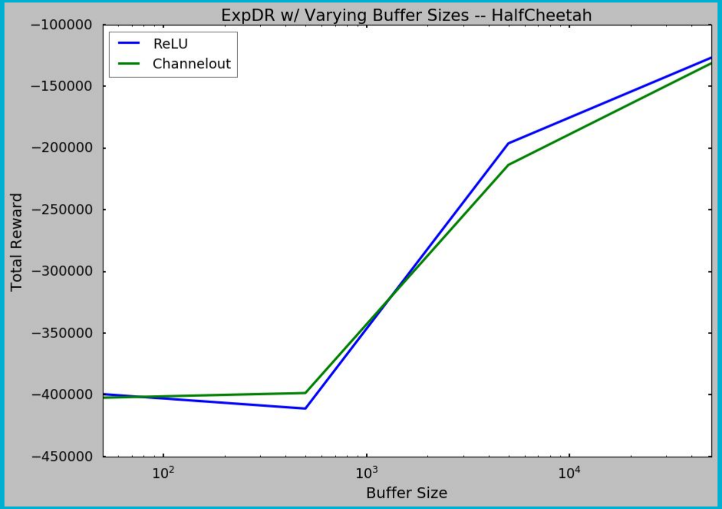

Distributional Regularizers -- Pendulum
Pct Non-Zero Activations
Episode
expDR w/ ReLU
expDR w/ Channelout
gaussianDR w/ ReLU
gaussianDR w/ Channelout


Distributional Regularizers -- HalfCheetah
Cumulative Reward
Episode
expDR w/ ReLU
expDR w/ Channelout
gaussianDR w/ ReLU
gaussianDR w/ Channelout

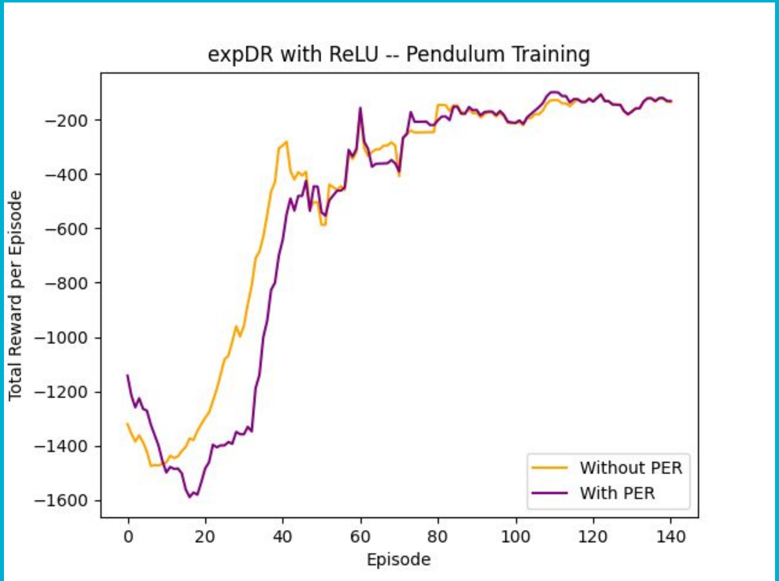

Distributional Regularizers -- HalfCheetah
Pct Non-Zero Activations
Episode
expDR w/ ReLU
expDR w/ Channelout
gaussianDR w/ ReLU
gaussianDR w/ Channelout

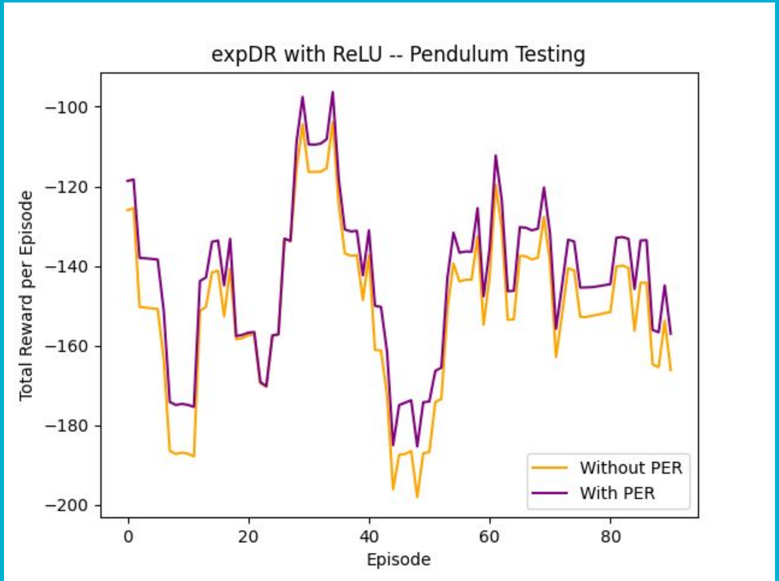
Without PER


ExpDR w/ Varying Buffer Sizes -- Pendulum
ReLU
Channelout
Total Reward
Buffer Size
−80000
−100000
−120000
−140000
−160000
−180000
−200000
10^2
10^3
10^4


ExpDR w/ Varying Buffer Sizes -- HalfCheetah
ReLU
Channelout
Total Reward
Buffer Size
−100000
−150000
−200000
−250000
−300000
−350000
−400000
−450000
10^2
10^3
10^4

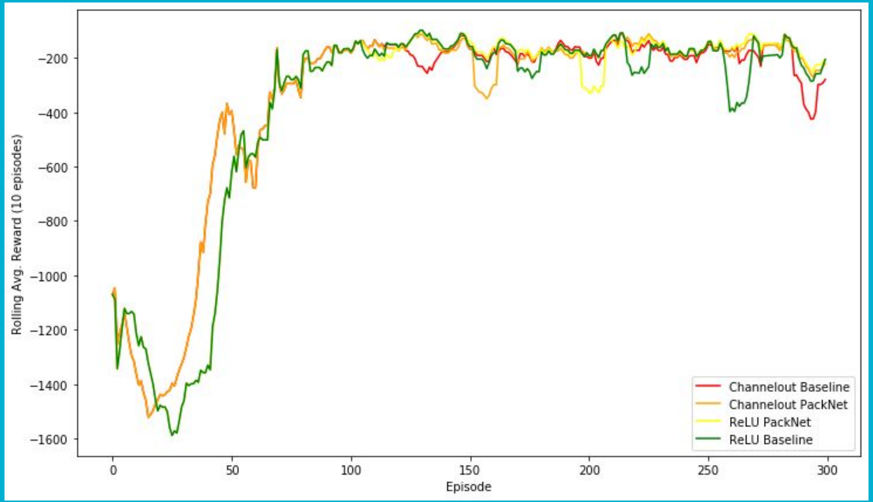
With PER


expDR with ReLU -- Pendulum Training
Total Reward per Episode
Episode
−200
−400
−600
−800
−1000
−1200
−1400
−1600
0
20
40
60
80
100
120
140
Without PER
With PER


expDR with ReLU -- Pendulum Testing
Total Reward per Episode
Episode
−100
−120
−140
−160
−180
−200
0
20
40
60
80
Without PER
With PER

PackNet Results

Trained over three random seeds of Pendulum, 100 episodes each.



Model	Mean Reward
Channel-out Baseline	-351.41
Channel-out PackNet	-344.58
ReLU PackNet	-366.8
ReLU Baseline	-378.31

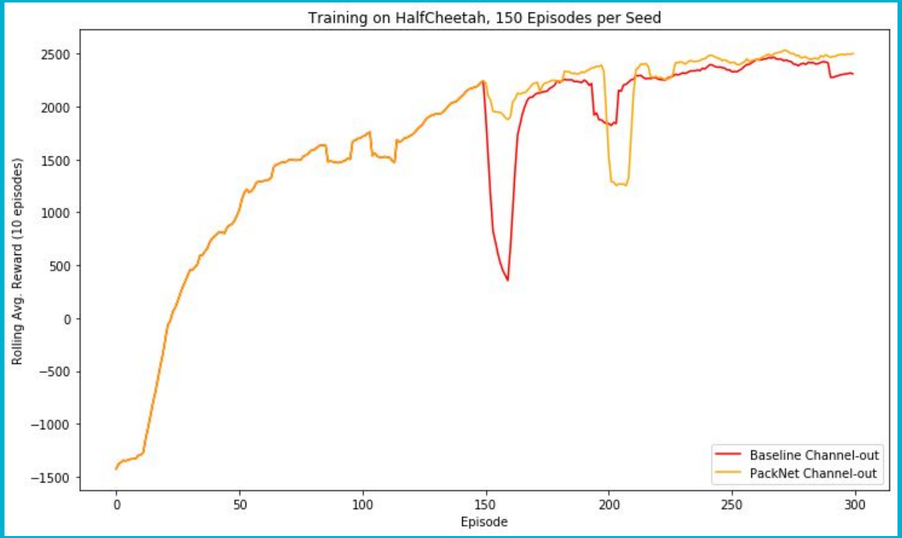
Testing over 100 episodes on the last seed and a completely new seed:

Model	Mean Reward
Channel-out Baseline (last)	-151.76
Channel-out Baseline (new)	-149.01
Channel-out PackNet (last)	**-135.91**
Channel-out PackNet (new)	-149.37
ReLU PackNet (last)	-151.03
ReLU PackNet (new)	-161.27
ReLU Baseline (last)	-157.56
ReLU Baseline (new)	-148.78

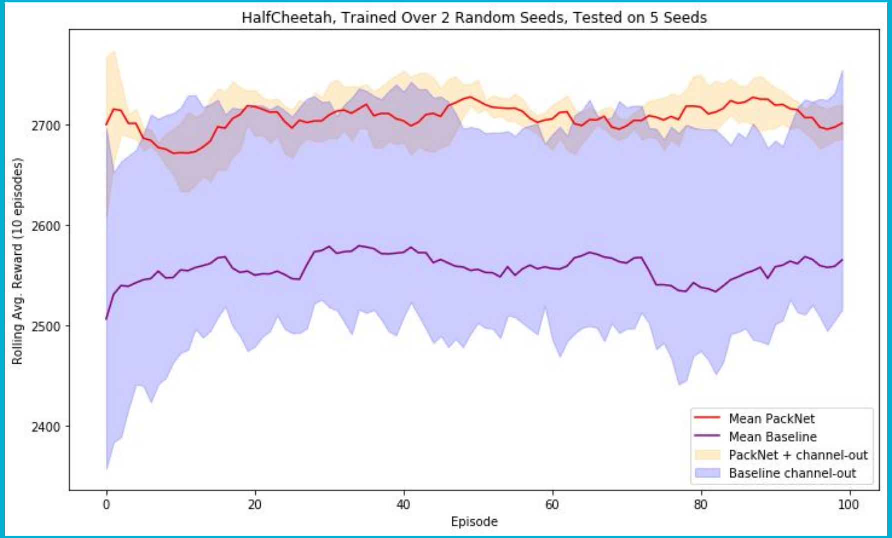
So PackNet, and especially with channel-out, works better than baseline. But these results are not very strong. More testing...

Channel-out Activation

Trained over two random seeds of HalfCheetah, 150 episodes each.



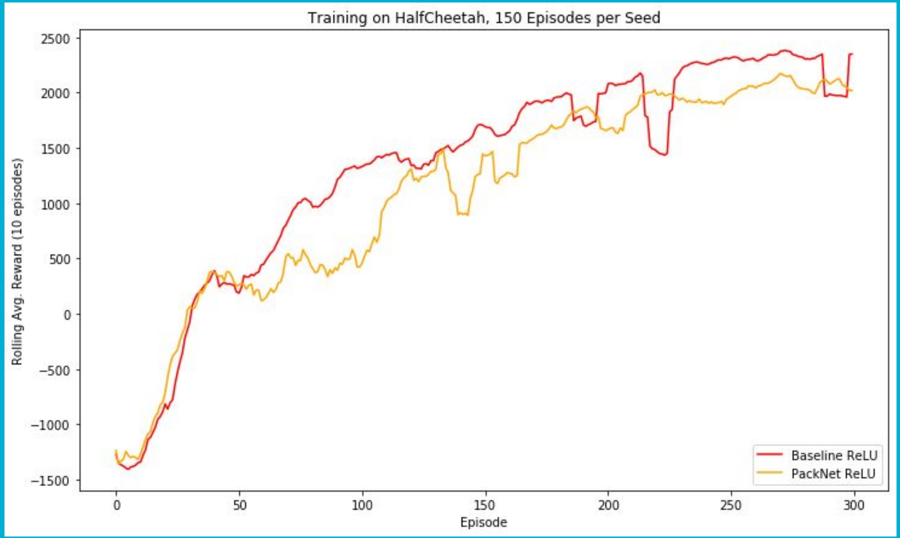
Testing on 100 episodes of 5 new random seeds.



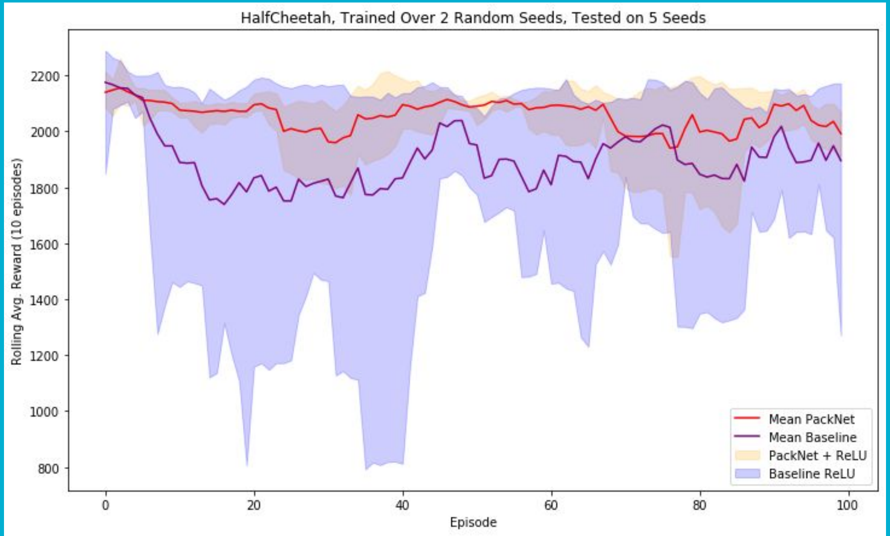
Stronger results! But is it PackNet that causes this success, or its combination with channel-out?

ReLU Activation

Trained over two random seeds of HalfCheetah, 150 episodes each.



Testing on 100 episodes of 5 new random seeds.



So it's probably the combination. We claim that channel-out encodes observation information into the network's pathways, and PackNet increases network capacity usage like dropout, but without the added variance.

Conclusion/Future Work

Main takeaways from Sparse Pathway Encoding:

Some activation function need dropout to be effective, but the variance effect of dropout is too great to make its use worthwhile with them.

SELU + alphaDropout shows it is possible to add a similar effect without using the vanilla notion of dropout.

Conclusion/Future Work

Main takeaway from Distributional Regularizers:

ExpDR + ReLU is the only Distributional Regularizer model that worked well from a rewards standpoint, though all implementations were successful in creating a sparse representation.

- Our hypothesis about GaussianDR + channel-out was wrong! (Not too surprising because channel-out can't necessarily guarantee zero-output like ReLU, at the pool-level only)
- GaussianDR + ReLU worked well for DQN, what gives?
    - Is it the algorithms themselves, or do discrete vs continuous action spaces interact with sparse representations differently? Need more investigation to find out.

Conclusion/Future Work

Main takeaways from PackNet:

PackNet and channel-out need each other.

- Channel-out captures observation information and encodes it into network pathways
- PackNet increases network capacity usage so the pathway information is more abundant and expressive.

PackNet can act as a limited alternative to dropout in Deep RL for some activation functions.

References

R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, “Measuring catastrophic forgetting in neural networks,” 2017.
I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, “Maxout networks,” 2013.
Q. Wang and J. JaJa, “From maxout to channel-out: Encoding information on sparse pathways,” 2013.
V. Liu, “Sparse representation neural networks for online reinforcement learning,” Master’s thesis, University of Alberta, 2019.
A. Mallya and S. Lazebnik, “Packnet: Adding multiple tasks to a single network by iterative pruning,” 2017.