A APPENDIX

A.1 DEMONSTRATING CORRECTNESS WITH THE NN-POWER METHOD

Before continuing further, it is important that we empirically verify Prop. 1 and determine it's numerical precision when used in practice. Unfortunately, when dealing with many small values, such as the second derivatives of an NN's parameters, minor errors in precision add up. This could make a mathematically viable method unusable in practice. Thus, we present a brief experiment which aim's to test the limits of the efficient Hessian-vector product. The power method is an algorithm for obtaining the top k eigenvalues of a matrix. The full algorithm is shown in Sec. A.3, however, the main approach is to use the recurrence relation of $b_{t+1} \leftarrow Ab_t/||Ab_t||$ where b is a randomly initialized N-dimensional vector and A is the $N \times N$ matrix that we aim to obtain the eigenvalues for.

In our experiment we implement the power method for the Hessian of an NN and obtain its top k eigenvalues. However, instead of performing the recurrence step of $b_{t+1} \leftarrow H(\theta)b_t/||H(\theta)b_t||$ we use $b_{t+1} \leftarrow \nabla_{\theta^*} bt^T \nabla_{\theta^*} \mathbb{E}_X[L(\theta^*)]/||\nabla_{\theta^*} bt^T \nabla_{\theta^*} \mathbb{E}_X[L(\theta^*)]||$, replacing the Hessian-vector product in exactly the same manner as in Prop. 1. We then compare the obtained eigenvalues to the real eigenvalues of the Hessian. Since this experiment requires the full Hessian and its eigenvalues to be explicitly calculated to obtain the ground truth values we are limited in the scale of this experiment. We use a simplification of MNIST where the network performs binary classification on the labels of < 5 and $>= 5$. While this is a simple task, it still involves a deep non-linear NN working on a fairly naturalistic setting. See Sec. A.4 for additional details on the network architecture. The comparison between the true and obtained eigenvalues can be seen in Fig. 5. We see that the obtained eigenvalues match the true eigenvalues almost exactly, in spite of the iterative nature of the algorithm and repeated application of the efficient Hessian-vector multiplication.

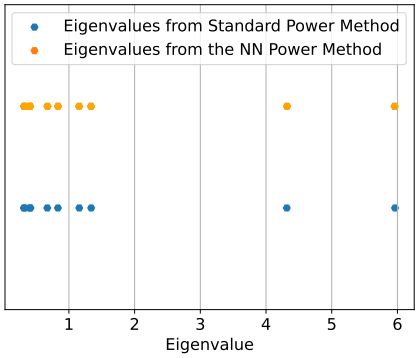

Figure 5: Comparison of the ground truth Eigenvectors of the Hessian matrix with those obtained from the NN Power Method

A.2 HESSIAN-VECTOR PRODUCT DERIVATION AND APPLICATION TO EWC

Here we derive the efficient Hessian-Vector product method and demonstrate its application to calculating the Full EWC update step. First the efficient Hessian-Vector product is derived as follows (where $\mathbf{X}$ is a general dataset):

$$
\begin{aligned}
H(\theta) &= \nabla_\theta \nabla_\theta L \in \mathbb{R}^{N \times N} \text{ and } V \in \mathbb{R}^N \text{ is some vector independent of } \theta \\
H(\theta)V &= \nabla_\theta \nabla_\theta L V \\
&= V^T \nabla_\theta \nabla_\theta L \\
&= \nabla_\theta (V^T \nabla_\theta L) \text{ since } V \text{ is independent of } \theta
\end{aligned}
$$

Likewise for the expected Hessian:

$$
\begin{aligned}
E_{\mathbf{X}}[H(\theta)] &= E_{\mathbf{X}}[\nabla_\theta \nabla_\theta L] \in \mathbb{R}^{N \times N} \text{ and } V \in \mathbb{R}^N \text{ is some vector independent of } \theta \\
E_{\mathbf{X}}[H(\theta)]V &= E_{\mathbf{X}}[\nabla_\theta \nabla_\theta L]V
\end{aligned}
$$

$$= V^T E_{\mathbf{X}}[\nabla_\theta \nabla_\theta L]$$

We can now use the Leibniz Integration rule to change the order of integration and differentiation:

$$= \nabla_\theta (V^T \nabla_\theta E_{\mathbf{X}}[L]) \text{ since } V \text{ is independent of } \theta$$

Note that V and $\nabla_\theta E_{\mathbf{X}}[L]$ are N-dimensional vectors and the dot-product $V^T \nabla_\theta E_{\mathbf{X}}[L]$ results in a scalar value before the second derivative is applied.

This can then be applied to calculate the Full EWC update step as follows:

$$\begin{aligned}
\nabla_\theta EWC(\theta) &= (\theta - \theta_A)^T I(\theta_A) \\
&= (\theta - \theta_A)^T \mathbb{E}[H(\theta_A)] \\
&= (\theta - \theta_A)^T \mathbb{E}[\nabla_{\theta_A} \nabla_{\theta_A} L(\theta)] \\
&= (\theta - \theta_A)^T \nabla_{\theta_A} \nabla_{\theta_A} \mathbb{E}[L(\theta)] \\
\nabla_\theta EWC(\theta) &= \nabla_{\theta_A} (\theta - \theta_{A_{const}})^T \nabla_{\theta_A} \mathbb{E}[L(\theta)]
\end{aligned}$$

A.3 Neural Network Power Method

In Alg. 1 we provide the algorithm used to obtain the spectrum of the NN Hessian matrix in Sec. A.3. This is a version of the Power Method designed specifically to work with the NN Hessian using the efficient Hessian-Vector product.

Algorithm 1 Neural Network Power Method

Require: K: the number of desired Eigenvalues
Require: $\mathbf{X}$: a dataset to obtain the NN's loss
  $Q \sim \mathcal{N}(0, \sigma^2)$ where σ^2 is a small constant value
  $diff \leftarrow \infty$
  while $diff > \epsilon$ where ϵ is a sufficiently small value **do**
    $Q_{prev} \leftarrow Q$
    $Z \leftarrow \mathbf{0}_{N \times K}$ where N is the number of network parameters
    for $j \leq K$ **do**
      $Z_{:,j} \leftarrow \nabla_\theta Q_{:,j}^T \nabla_\theta \mathbb{E}_X[L(\theta)] / ||\nabla_\theta Q_{:,j}^T \nabla_\theta \mathbb{E}_X[L(\theta)]||||$
    end for
    $Q, R \leftarrow QR(Z)$
    $diff \leftarrow \sum_{i,j=0}^{N,N} (Q^{ij} - Q_{prev}^{ij})^2$
  end while
  return $|\mathbf{diag}(R)|, Q$

A.4 PERMUTED MNIST EXPERIMENTAL DETAILS

Here we will describe the details of the NN architecture and hyper-parameters used for the Permuted MNIST continual learning task. The set of hyper-parameters and implementation details are summarized in Tab. 1 and the network architecture is shown in Fig. 6.

Table 1: Hyper-parameters and implementation details for the Permuted MNIST task

Hyper-parameter	**Value**
Activation	ReLU
Learning Algorithm	Stochastic (Batch) Gradient Descent
Step Size	0.01
Full EWC Regularization Rate	1e-3
Diagonal EWC Regularization Rate	1e2
Batch Size	128
Learning Regime Initialization Variance	0.005
Lazy Regime Initialization Variance	0.1

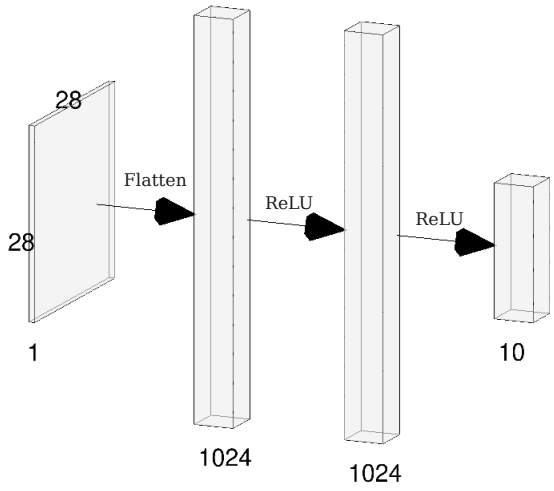

Figure 6: Network architecture for the Permuted MNIST task

A.5 PROOF OF PROP. 2

In this section we provide the proof of Bayes optimality of Full EWC described in Prop. 2 which states:
Assuming:

1. θ_A minimizes the expected loss $\mathbb{E}_{D_A}[L(\theta)]$ and that $\mathbb{E}_{D_A}[L(\theta_A)] = 0$.
2. Constant covariance ($\Sigma(\theta) = \Sigma$) in the region of the MAP estimator in parameter space: $\hat{\theta}$.
3. Mean parameter values which are independent of all other parameters: $\frac{\partial \mathbb{E}[\theta_i]}{\partial \theta_j} = 1$ if $i = j$ and 0 otherwise.

then Full EWC approximates the Bayes Optimal (Minimum Means Squared Error) estimator.
This proof technique is based on a similar result which has been shown for linear regression (Advani & Ganguli, 2016) and uses the Laplace approximation for neural networks from Zhang et al. (2018). For this proof there are three points of consideration in parameter space: θ_A is the optimal parameter values at the end of training the previous task. It is the mean of the Gaussian prior distribution below. Then, θ are actual values of the NN's parameters while training the current task. Finally, $\hat{\theta}$ is the MAP parametrization for the NN on the current task of learning dataset X. In using the Laplace method below, we are required to sum over all reasonable network parametrizations of

the NN. We denote one possible MAP parametrization in this set of possible MAP estimators as $\hat{\theta}_q$.
We begin by defining the likelihood and prior distributions:

$$P(X|\theta) = \exp(\log(P(X|\theta)))$$
$$P(\theta) = \frac{1}{\sqrt{(2\pi)^N|\Sigma|}} \exp\left(-\frac{\beta}{2}(\theta_A - \theta)^T\Sigma^{-1}(\theta_A - \theta)\right)$$

Using Bayes Theorem this allows us to derive the posterior distribution for the parameter space:

$$P(\theta|X) = \frac{1}{Z}\exp(\log(P(X|\theta)))\frac{1}{\sqrt{(2\pi)^N|\Sigma|}} \exp\left(-\frac{\beta}{2}(\theta_A - \theta)^T\Sigma^{-1}(\theta_A - \theta)\right)$$
$$= \frac{1}{\sqrt{(2\pi)^N|\Sigma|}} \exp\left(\log(P(X|\theta)) - \frac{\beta}{2}(\theta_A - \theta)^T\Sigma^{-1}(\theta_A - \theta)\right)$$

By using this posterior and integrating over the parameter space we can obtain the Bayes estimator:

$$f^*(\theta) = \int f(\theta)\frac{1}{Z\sqrt{(2\pi)^N|\Sigma|}} \exp\left(\log(P(X|\theta)) - \frac{\beta}{2}(\theta_A - \theta)^T\Sigma^{-1}(\theta_A - \theta)\right) d\theta$$

To evaluate this integral we use Laplace method around the MAP estimators: $\hat{\theta}_q$. For this to be valid
we require the density to concentrate around the MAP estimator. To achieve this we can increase the value of β

$$= \frac{1}{Z\sqrt{(2\pi)^N|\Sigma|}} \sum_{\hat{\theta}_q} \exp\left(\log(P(X|\hat{\theta}_q)) - \frac{\beta}{2}(\theta_A - \hat{\theta}_q)^T\Sigma^{-1}(\theta_A - \hat{\theta}_q)\right)$$
$$\int f(\theta)\exp\left((\hat{\theta}_q - \theta)^T(\nabla_\theta\nabla_\theta\log(P(X|\theta)) - \frac{\beta}{2}\Sigma^{-1})(\hat{\theta}_q - \theta)\right) d\theta$$

Using Assumption 1 we get that the Hessian matrix is equal to the negative FIM: $\nabla_\theta\nabla_\theta\log(P(X|\theta)) = -I(\theta)$
Using Assumptions 2 and 3 we get that the inverse covariance matrix is equal to the FIM: $\Sigma^{-1} = I(\theta)$
By substituting both results into the Bayes estimator we will incur error at all points in the loss landscape except at the MAP estimator. The error from replacing the Hessian with the FIM is on the order of $\mathbb{E}\nabla_{\theta_i}L(\theta)\nabla_{\theta_j}L(\theta)$
This error will be small in the region of the MAP. Thus by concentrating the density to perform the Laplace method we also can ensure the error incurred by substituting in the FIM is minimal. Similarly, we restrict the density to regions where we have assumed the variance is constant. Thus replacing the inverse covariance is valid due to the concentrated density

$$f^*(\theta) = \frac{1}{Z\sqrt{(2\pi)^N|\Sigma|}} \sum_{\hat{\theta}_q} \exp\left(\log(P(X|\hat{\theta}_q)) - \frac{\beta}{2}(\theta_A - \hat{\theta}_q)^T I(\theta)(\theta_A - \hat{\theta}_q)\right)$$
$$\int f(\theta)\exp\left(-\frac{1+\beta}{2}(\hat{\theta}_q - \theta)^T I(\theta)(\hat{\theta}_q - \theta)\right) d\theta$$

Since we have assumed constant covariance in the area of the MAP, this means the FIM is locally constant.
Using this fact we solve the Gaussian integral and obtain:

$$f^*(\theta) = \frac{1}{|I(\theta)|} \sum_{\hat{\theta}_q} \exp\left(\log(P(X|\hat{\theta}_q)) - \frac{1}{2}(\theta_A - \hat{\theta}_q)^T I(\theta)(\theta_A - \hat{\theta}_q)\right) f(\hat{\theta}_q)$$

This is a weighted average of all high probability estimators locally in the region of the prior mean θ_A
When there is one local high probability estimator then the Bayes optimal estimator is also the MAP estimator:

$$f^*(\theta) = f(\hat{\theta})$$

which is found with Full EWC by optimizing:

$$L(\theta) = \log(P(X|\hat{\theta})) - \frac{1}{2}(\theta_A - \hat{\theta})^T I(\theta)(\theta_A - \hat{\theta})$$

A.6 RL EXPERIMENT DETAILS

Finally, we now provide the implementation details and hyper-parameters (Tab. 2) and NN architecture (Fig. 7) for the Reinforcement Learning task on the Boxman domain.

Table 2: Hyper-parameters and implementation details for the Permuted MNIST task

Hyper-parameter	**Value**
Activation	ReLU
Learning Algorithm	DDQN
Step Size	1e-3
Discount Factor	0.99
Full EWC Regularization Rate	5e-1
Diagonal EWC Regularization Rate	1e1
Batch Size	64
Replay Buffer Size	1e5
Initialization Variance	0.1

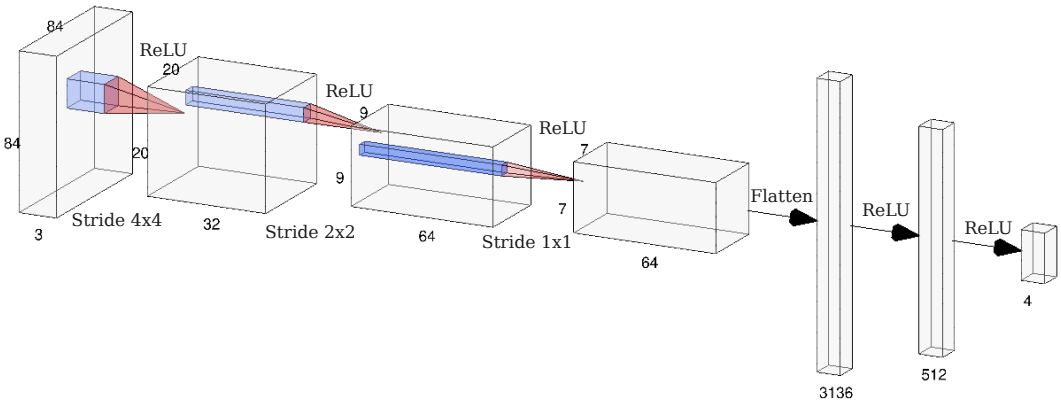

Figure 7: Network architecture for all of the Reinforcement Learning experiments

A.7 ADDITIONAL REINFORCEMENT LEARNING EXPERIMENTS

A.7.1 SMALLER ATARI MODEL

In this experiment we evaluate the performance of the various EWC regularizers on a network with half the number of convolutional and dense neurons than the architecture used in Sec. 5. The architecture used in Sec. 5 is the largest we could feasible use (and the architecture and hyper-parameters are depicted in Sec. A.6) but we suspect that this network is not sufficiently over-parametrized to fit the conditions for EWC to be most useful. Thus, this experiment firstly serves to determine if a performance drop is incurred when making the architecture even smaller. This is clearly seen in the results shown in Fig. 8 (left) and Fig. 8 (right). This is at least an indication that the architecture of Sec. 5 is not over-parametrized and does not have significant excess capacity. Interestingly, we see that Full EWC alone offers little benefit towards avoiding forgetting - there is a minor effect seen when switching from Pong to Breakout where some Pong performance is maintained at the small expense of Breakout. Combined EWC, however, does slightly out-perform Diagonal EWC by remembering Pong and allowing for some new learning on Space Invaders (it completely fails on Breakout). Diagonal EWC learns Breakout, however but forgets Pong and learns Space Invaders to a similar degree as Combined EWC. Thus, we once again see the effect that by incorporating the full FIM EWC is better able to maintain performance over longer task sequences. This, however, comes at the expense of new learning particularly when the network is parameter constrained. Finding this salient and consistent effect from incorporating the full FIM is an important result, however, this also demonstrates our point in Sec. 5 and motivates a degree of caution in drawing conclusions on the best version of EWC for RL in ideal conditions.

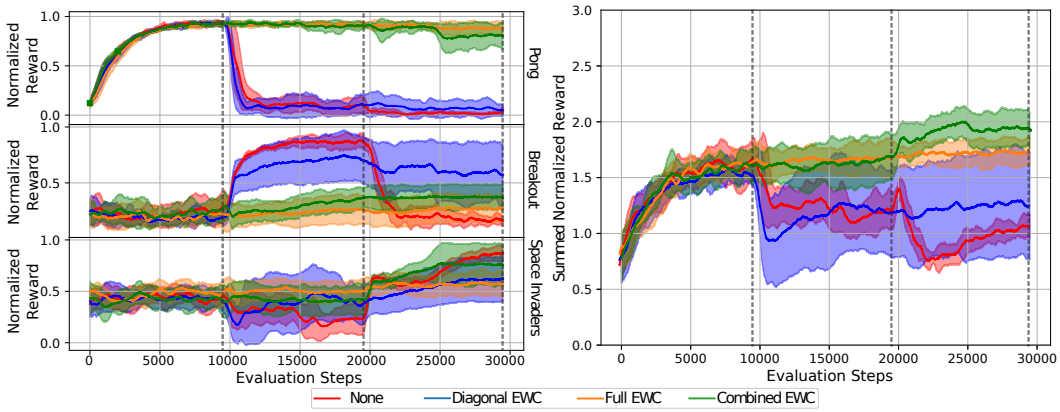

Figure 8: Results of the various EWC models trained to play three Atari domains (Pong, Breakout, Space Invaders) in order. (left) Mean standardized rewards in the three domains separately over the number of evaluation steps made by the agent. (right) Mean sum of the standardized rewards in the three domains over the number of evaluation steps from the agent. This demonstrates the aggregate performance of the model across all tasks. Once again we see a trade-off between the long-range memory of Full EWC and the ability to learn new tasks of Diagonal EWC. Importantly, Combined EWC can demonstrate both positive qualities to some degree - mitigating the trade-off.

A.7.2 BOXMAN EXPERIMENT

Our final experiment aims to establish the detriment of sharing input and outputs spaces between tasks for EWC. For this we use the Boxman continual learning domain, where agents are tasked with collecting a subset of objects from a 2d environment. One such environment is shown in Fig. 9. The NN receives pixel inputs of the environment and must navigate towards retrieving the objects by outputting one of the four cardinal directions. The objects in the environment are moved to new locations after every trajectory, thus, the agent cannot merely memorize their locations. To make this a continual learning task, the agent is first trained to collect all blue objects, then beige objects and finally purple objects. As predicted by prior work (Lee et al., 2022) the fact that the environment is very similar between tasks will make the continual learning far more difficult than if completely different learning tasks were used. This is due to the NN aiming to reuse the features it has learned on previous tasks to complete the new tasks, thus, interfering with the previous task. Finally, we display the colour of the objects the agent should obtain in the top left corner of the input image. Without this the agent would be mapping the same states to different outputs after every task switch and continual learning would be impossible. Additionally, we maintain a replay buffer between tasks. As mentioned in Sec. 2.1 this alone does help continual learning, but also provides some signal to the agent to learn to use the task descriptor at the top left. Without this replay buffer continual learning is also unlearnable, since there is no signal to use the task descriptor otherwise. We hyper-parameter tuned the buffer size to allow for learning but also for a baseline agent (with no EWC) to forget prior tasks. Thus, we are still able to determine the impact on EWC of performing continual learning in very similar or shared domains.

The result of this experiment are shown in Fig. 10 (right) which displays the average sum of rewards across the three tasks as learning occurs (all tasks produce a maximum reward of 1.0 after each trajectory). The average is taken over 100 timesteps. We do see a slight benefit at the end of training from using Full and Combined EWC. However, in the case of Full EWC this appears to come at the expense of slower learning and worse performance throughout training. Overall we only see very modest improvements in summed rewards after step 500 (after the first task switch) which indicates that the models are all forgetting prior tasks. This is also apparent as there are initial dips in performance at step 500 and 1000 when task switches occur. Most importantly, none of the models perform statistically significantly better than any others. In other words, in this domain EWC is no more beneficial than training without it. This supports the notion that tasks with shared input and action spaces pose particular problems for continual learning and EWC. Additionally, a similar effect is likely to occur when switching from Pong to Breakout in Sec. 5.

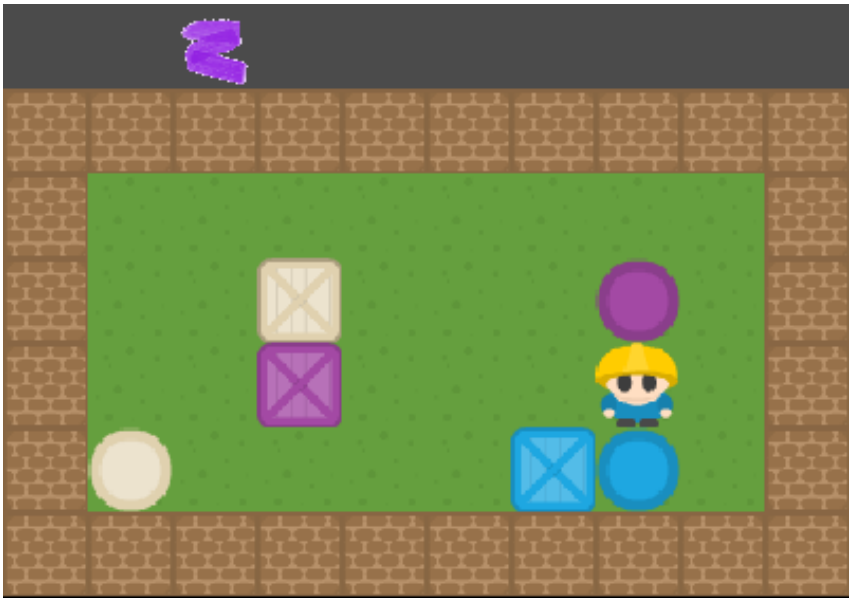

Figure 9: Illustration of the object collection domain with 84*84*3 pixel observations. The character sprite represents the agent and the top row icon represents the task.

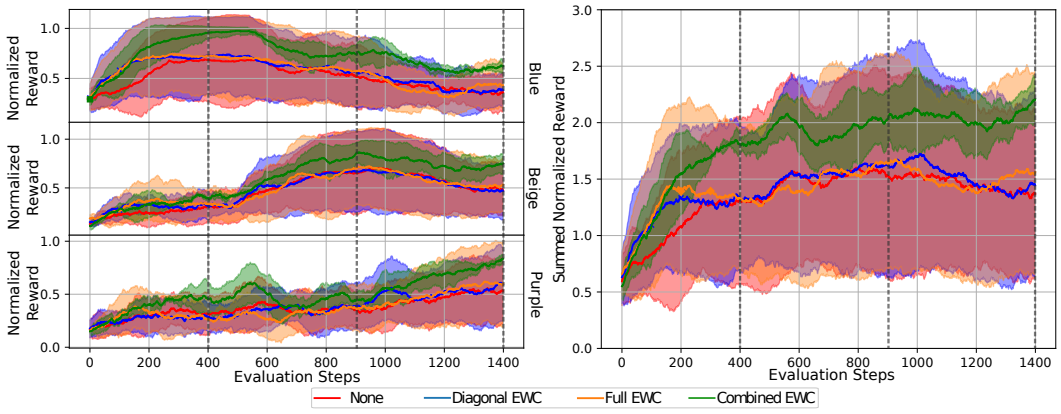

Figure 10: Results of the various EWC models trained to collect blue, beige and purple objects in sequence in the Boxman domain. (left) Mean standardized rewards in the three tasks separately over the number of evaluation steps made by the agent. (b) Mean sum of the standardized rewards in the three domains over the number of evaluation steps from the agent. This demonstrates the aggregate performance of the model across all tasks. All models struggle to continually learn in this domain, and EWC does not outperform the vanilla Double DQN. We do see a slight improvement from Combined EWC and an increase in the consistency of its performance over the other models.