Progress & Compress: A scalable framework for continual learning. Supplementary material

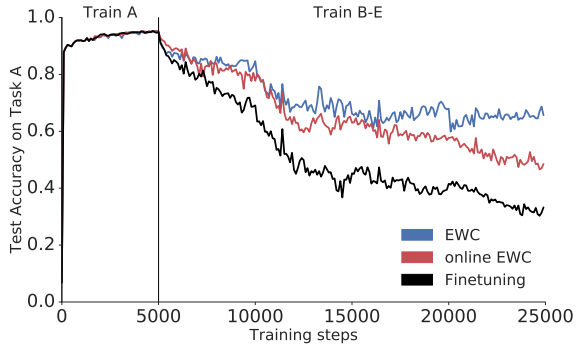


Figure 1. Performance retention on permuted MNIST. Shown is the test accuracy on an initial permutation (Task A) over the course of training on the remaining set of tasks (Tasks B-E).

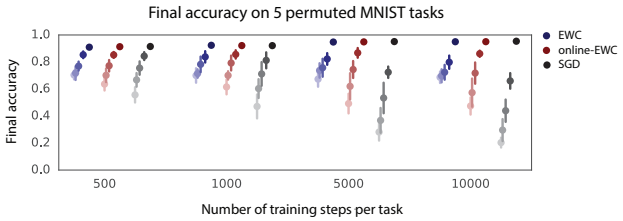


Figure 2. Comparison of EWC, online-EWC and Finetuning We ran the three training methods on 5 permuted-MNIST tasks ((Kirkpatrick et al., 2017)). The accuracy at the end of training is shown for each task with the fainter colours relating to the older tasks. The number of training steps on the x-axis relates to the number of minibatches of each task used for the training. In this regime (see text for details), EWC appears to be a better choice for a small number of training steps.

A. Retention of task performance for EWC and online EWC

The difference between EWC and online-EWC is in their weighting of the past experiences, with EWC putting more weight on the initial tasks and online-EWC favouring the most recent past. For an optimal setting, where the optimisation converges and all penalty terms can be satisfied (Huszár, 2017), online-EWC is often a better choice. However, it is likely that in difficult problems, the network (an agent) doesn't get enough time/training data to arrive at the optimal solution.

We investigated this hypothesis in a series of experiments with a sequential learning of permuted-MNIST images, similar to the experiments shown in (Kirkpatrick et al., 2017). In order to emulate learning difficult problems, we have not optimised the hyper-parameters, nor used any dropout or early stopping. Instead, we used a small MLP (layers consisting of 30-30-10 neurons, and Relu nonlinearities between the first two).

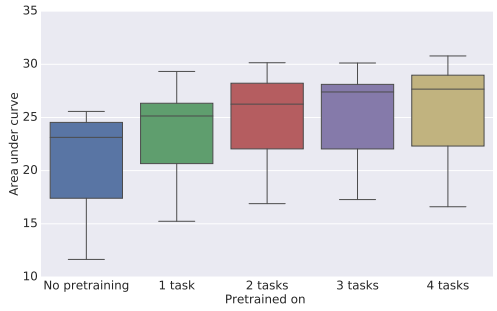
Figure 1 demonstrates the retainment of the skill for the initial task (Task A) by EWC, online-EWC and pure SGD training (with no additional penalties), over the course of learning on a total of 5 permutations (Tasks A-E). As expected, EWC keeps higher accuracy for Task A.

In Fig. 2, we plot the final accuracy for each of the tasks (colour saturating from the faintest one representing Task A to the fully saturated for Task E), as a function of the number of training steps spent on each task. Here, we run 10 training sessions per fixed amount of training steps, generating new permutations for each training, but feeding exactly the same data to all methods (dot represents the mean and bars: 1 standard deviation (bar)).

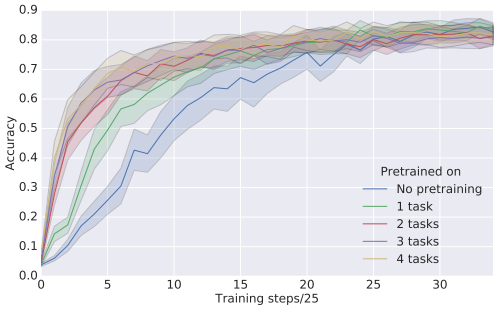
For a small number of training steps (500 and 1000, training over minibatches of size 32), the network benefits from holding on to the memories of the earlier tasks (the accuracy of EWC, i.e. all blue dots in the plot are higher than for the online EWC, the red dots). With more data (10,000 training steps), holding on to the initial parameters makes it more difficult to retain the most recent tasks (compare the dark blue dots of n=10,000 with n=500). In this example (with a relatively high learning rate $\eta = 0.1$), the online EWC doesn't seem to find a good balance between the loss and penalties and the performance on older tasks is not well retained (faint red dots), although it's still better than using no penalty at all (grey dots).

B. Faster learning on Omniglot

While the experiments on Omniglot in the main text show that all considered methods fail to obtain a higher accu-



(a) Area under the learning curve



(b) Averaged learning curves

Figure 3. Faster learning on Omniglot with Progress & Compress. Results averaged over 10 alphabets.

racy through sequential learning in the Omniglot case, improved data efficiency can indeed be observed for Progress&Compress (P&C).

In order to test this in isolation, we trained P&C (with online-EWC) on 10 unique Omniglot alphabets, after having learned up to 4 different tasks. We show both the averaged learning curves as well as the area under those learning curves in Figure 3. The results suggest that pre-training on a small number of tasks can greatly improve data efficiency, with this effect plateauing for more than 4 tasks.

C. Experiment details

C.1. Omniglot

In the Omniglot experiments, we used a convolutional network similar to the one introduced by Vinyals et al. (2016), ensuring each method has sufficient capacity to learn all tasks. Namely, the network consists of 4 blocks of 3×3 convolutions with 64 filters followed by a ReLU nonlinearity, and 2×2 max-pooling before predicting class probabilities. In the case of P&C and Progressive Nets, all network columns follow this architecture. As suggested in (Rusu et al., 2016), non-linear adapters for convolutional networks are implemented by replacing each linear layer by a 1×1 convolutions using an identical number of filters.

Similar to the setup proposed in (Koch et al., 2015) we used a 60/20/20% split to obtain train-/valid- and test-sets. In addition, we rescaled all images to 28×28 and augment the dataset by including 20 random permutations (rotations and shifting) for each image. Note that since we are not treating Omniglot in the usual few-shot learning fashion, we do not distinguish between train and test alphabets.

For all considered models, we used a batch size of 32 and perform 2500 training updates with Stochastic Gradient Descent and a fixed learning rate of 0.1 (0.05 during distillation), which we found sufficient to learn each alphabet separately from scratch.

For EWC, online EWC and P&C, we chose the regularisation strength λ and forgetting coefficient γ by running a grid search for λ=[10.0, 12.5, 15.0, 17.5, 20.0, 22.5, 25.0] and γ=[0.7, 0.8, 0.9, 0.95, 0.99]. For Learning Without Forgetting (LwF) we tried λ=[0.05, 0.1, 0.15, 0.2, 0.25, 0.3]. For distillation within P&C and LwF, we found a softmax temperature $\tau = 2.0$ to work best. All hyperparmeters were tuned by maximising the averaged performance over all tasks using aforementioned validation set.

Note that we use the same network and optimisation settings throughout all experiments. This is with the exception of results showing positive transfer and forgetting in isolation, in which case we fix λ for all EWC methods to provide a fair comparison.

C.2. Atari & Navigation tasks

For both Atari&Navigation tasks we use the same network as in (Mnih et al., 2013), adopting it to actor-critic algorithms by estimating both value and policy through linear layers connected to the final output of a shared network. During optimisation, we use a batchsize of 20, unroll length of 20 and update the model parameters with RMSProp (using $\epsilon = 0.1$), linearly annealing the learning rate to 0 over the course of training. For navigation mazes, we used an initial learning rate of $\alpha = 0.004$ and entropy cost $\beta = 0.003$. For Atari games, we used $\alpha = 0.0006$ and $\beta = 0.01$. In both cases, we receive RGB environment frames as $84 \times 84 \times 3$ tensors. As is common, we apply each action 4 times to the environment.

Furthermore, we use clip rewards so that the maximum absolute reward is 1.0. We also use a baseline cost of 0.5 in the policy gradient loss. The discounting factor was set to 0.99.

EWC was separately tuned choosing λ from [500, 1000, 1500, 2000, 2500, 3000]. As the scale of the losses differ, we selected λ for online EWC as applied in P&C among [25, 75, 125, 175]. We use 100 minibatches of equal size to

estimate the diagonal Fisher.

References

Huszár, F. On quadratic penalties in elastic weight consolidation. *arXiv preprint arXiv:1712.03847*, 2017.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, pp. 201611835, 2017.

Koch, G., Zemel, R., and Salakhutdinov, R. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pp. 3630–3638, 2016.