# Appendices to "A General Framework for Continual Learning of Compositional Structures"

by **Jorge A. Mendez** and **Eric Eaton**

## A. Experimental setting

Below, we give details of our experimental setting.

### A.1. Data Sets

**Linear models** We used three data sets for experimenting with the linear models that have previously been used for evaluating linear continual learning models (Ruvolo & Eaton, 2013). The **Landmine** data set consists of $T_{\mathsf{max}} = 29$ tasks, each of which requires detecting land mines in radar readings from a different geographical region. The Facial Recognition (**FERA**) data set tasks involve recognizing one of three facial expressions for one of seven individuals, for a total of $T_{\mathsf{max}} = 21$ tasks. Finally, the London Schools (**Schools**) data set contains $T_{\mathsf{max}} = 139$ tasks, each corresponding to exam score prediction in a different school. For these three data sets, we used the same data pre-processing and train/test split used by Ruvolo & Eaton (2013).

**Deep models** We used five data sets for evaluating our deep models. Binary MNIST (**MNIST**) is a common benchmark for continual learning algorithms, where each task is a binary classification problem between a pair of digits. We constructed $T_{\mathsf{max}} = 10$ tasks by randomly sampling the digits, allowing digits to be reused across tasks. The Binary Fashion MNIST (**Fashion**) data set is equivalent in format to MNIST, but labels correspond to items of clothing. For these two data sets, all models used a task-specific input transformation layer $\mathcal{E}^{(t)}$ initialized at random and kept fixed throughout training, to ensure that the input spaces were sufficiently different (Meyerson & Miikkulainen, 2018). A more complex continual learning problem commonly used in the literature is Split CUB-200 (**CUB**), where the agent must classify bird species. We created $T_{\mathsf{max}} = 20$ tasks by randomly sampling ten species for each, without replacement across tasks. CUB images were cropped by the bounding boxes available with the data set, and resized to $224 \times 224$, and all agents used a frozen ResNet-18 pre-trained on ImageNet as a feature extractor shared across all tasks. For these first three data sets, all architectures were simple fully-connected networks. To show that our framework supports more complex convolutional architectures, we used two additional data sets. We constructed a continual learning version of CIFAR-100 (**CIFAR**) with $T_{\mathsf{max}} = 20$ tasks by randomly sampling five classes per task, without replacement across tasks. Finally, we used the **Omniglot** data set, which consists of $T_{\mathsf{max}} = 50$ multi-class classification problems, each corresponding to detecting handwritten symbols in a given alphabet. The inputs to all architectures for CIFAR and Omniglot were the images directly, without any transformation $\mathcal{E}^{(t)}$. Details are summarized in Table A.1. For MNIST, Fashion, CUB, and CIFAR we used the standard train/test split, and further divided the training set into 80% for training and 20% for validation. For Omniglot, we split the data into 80% for training, 10% for validation, and 10% for test, for each task. Validation sets were only used by dynamic + compositional learners for selecting whether to keep a new components.

Details are summarized in Table A.1.

### A.2. Network Architectures

We used $k = 4$ components for all compositional algorithms with fixed $k$. This is the only architecture parameter for linear models.

We based our soft layer ordering architectures on those used by Meyerson & Miikkulainen (2018), whenever possible. For MNIST and Fashion, we used a random and fixed linear input transformation $\mathcal{E}^{(t)}$ for each task, and each component was a fully-connected layer of 64 units. For CUB, all tasks shared a fixed ResNet-18 pre-trained on ImageNet[1] to extract features, followed by a task-specific input transformation $\mathcal{E}^{(t)}$ given by a linear trained layer, and each component was a fully-connected layer of 256 units. For CIFAR, there was no input transformation, and each component was a convolutional layer of 50 channels with $3 \times 3$ kernels and padding 1, followed by a max-pooling layer of size $2 \times 2$. Finally, for Omniglot, there was also no input input transformation, and each component was a convolutional layer of 53 channels with $3 \times 3$ kernels and no padding, followed by max-pooling of $2 \times 2$ patches. The input images to the convolutional nets in CIFAR and Omniglot were padded with all-zero channels in order to match the number of channels required by all component layers (50 and 53, respectively). All component layers were followed by ReLU activation

[1]The pre-trained ResNet-18 is provided by Pytorch, and we followed the pre-processing recommended at https://pytorch.org/docs/stable/torchvision/models.html.

Table A.1: Data set details summary.

| | FERA | Landmine | Schools | MNIST | Fashion | CUB | CIFAR | Omniglot |
|---|---|---|---|---|---|---|---|---|
| tasks | 21 | 29 | 139 | 10 | 10 | 20 | 20 | 50 |
| classes | 2 | 2 | — | 2 | 2 | 10 | 5 | 14–55 |
| features | 100 | 9 | 27 | 784 | 784 | 512 | 32×32×3 | 105×105 |
| feat. extract. | PCA | — | — | — | — | ResNet-18 | — | — |
| train | 225–499 | 222–345 | 11–125 | ∼9500 | ∼9500 | ∼120 | ∼2000 | 224–880 |
| val | — | — | — | ∼2500 | ∼2500 | ∼30 | ∼500 | 28–110 |
| test | 225–500 | 223–345 | 11–126 | ∼2000 | 2000 | ∼150 | 500 | 28–110 |

and a dropout layer with dropout probability $p = 0.5$. The output of each network was a linear task-specific output transformation $\mathcal{D}^{(t)}$ trained individually on each task. The architectures for jointly trained baselines were identical to these, and those for no-components baselines had the same layers but no mechanism to select the order of the layers.

### A.3. Algorithm Details

All agents trained for 100 epochs on each task, with a mini-batch of size 32. Compositional agents used the first 99 epochs solely for assimilation and the last epoch for accommodation (1). Dynamic + compositional agents followed this same process, but every assimilation step was concurrently done with and without addition of a new component; after the accommodation (1) step, the agent kept the new component if its validation performance with the added component represented at least a 5% relative improvement over the performance without the additional component. Joint agents trained all components and the structure for the current task jointly during all 100 epochs, keeping the structure for the previous tasks fixed, while no-components agents trained the whole model at every epoch.

ER-based algorithms used a replay buffer of a single mini-batch per task. Similarly, EWC-based algorithms used a single mini-batch to compute the approximate Fisher information matrix required for regularization, which we approximated as a Kronecker-factored matrix (Kirkpatrick et al., 2017; Ritter et al., 2018), and used a fixed regularization parameter $\lambda = 10^{-3}$.

To ensure a fair comparison, all algorithms, including our baselines, used the same initialization procedure by training the first $n_{\text{init}} = 4$ tasks jointly, in order to encourage the network to generalize across tasks. For soft ordering networks, the order of modules for the initial tasks was initialized as a random one-hot vector for each task at each depth, ensuring that each component was selected at least once. The structures over initial tasks were kept fixed during training, modifying only the weights of the components.

## B. Number of learned components

In our experiments, it was in many cases necessary to incorporate an accommodation (2) step in order for our algorithm to be sufficiently flexible to handle the stream of incoming tasks. This accommodation (2) step enables our methods to dynamically add new components if the existing ones are insufficient to achieve good performance in the new task. Table B.2 shows the number of components learned by each dynamic algorithm using soft layer ordering, averaged across all ten trials. Notably, in order for our methods to work on the CIFAR data set, they required learning almost one component per task. This explains why compositional algorithms without dynamic component additions were incapable of performing well on CIFAR. Moreover, FM adds far more components than other methods for MNIST and Fashion, which was the only way it was able to achieve high performance without adapting existing components.

Table B.2: Number of learned components. Standard errors reported after the ±.

| Base | MNIST | Fashion | CUB | CIFAR | Omniglot |
|---|---|---|---|---|---|
| ER | 5.2±0.3 | 4.9±0.3 | 5.9±0.3 | 19.1±0.3 | 9.3±0.3 |
| EWC | 5.0±0.3 | 4.7±0.2 | 5.7±0.3 | 19.5±0.2 | 10.1±0.4 |
| VAN | 5.0±0.2 | 4.8±0.3 | 6.1±0.3 | 17.7±0.3 | 10.0±0.7 |
| FM | 10.0±0.0 | 8.8±0.2 | 6.5±0.4 | 19.1±0.4 | 10.2±0.6 |