A APPENDIX

A.1 IMPLEMENTATION DETAILS-RAINBOWMNIST

All the images from Rainbow-MNIST are in 28×28 pixel resolution and with 3 channels. We used a 4 layer convolutional neural network for these experiments, with each layer having [32,32,64,64] number of filters. After every convolution ReLU activation and max pooling is applied to the features. At the last layer we take an average pooling. Finally, 64 dimensional features are passed through fully connected layer with a sigmoid activation. Since the Rainbow-MNIST uses MNIST data to classify, the final layer has 10 neurons, and the objective is to correctly classify the digits of different scales and colors into 10 classes (actual numbers irrespective of the rotation and background color). All the models were trained with 50 gradient updates.

A.2 IMPLEMENTATION DETAILS-ONLINE-CIFAR100

All the images from Rainbow-MNIST are in 32×32 pixel resolution and with 3 channels. We used a similar 7 layer convolutional neural network in a siamese network architecture for these experiments, with each layer having [32,32,32,64,64,64,128] number of filters. After every convolution ReLU activation and batchnorm is applied to the features. Every other layers have max-pooling. At the last layer we take an average pooling. The L2 distance between two images are calculated and a fully connected layer is used to classify the two images as same class or not.

A.3 BASELINE METHODS

TFS: Every time this model sees a new task (This needs to know the task boundary), the model resets to new sets of random weights. After that the model is trained only using the data from the new task. At the same time, we also rest the optimizer state to remove any effect of momentum from previous updates. The model is trained with Adam optimizer with a learning rate of 0.001.

TOE: Although TOE optimize the parameters on the all available task, training the model using all the available data is practically impossible. Therefore, we sample datapoints from the buffer and update the model parameters. However, since the number of data is increasing over time, the model also needs be updated on a good representative data of the online stream. Therefore, we increase the number of gradient updates over time. For ONLINE-CIFAR100 experiment, the number of gradients updates are increased by 10 for every 100 tasks. The model is trained with Adam optimizer with a learning rate of 0.001.

FTL: This method, first pretrains a model on the past data, and then fine-tune it on the very recent samples. Similar to TOE, we first train a set of weights using the datapoints in the data buffer. After that, the model is fine-tuned on the correct task data. However, after this adaptation and evaluation the fine-tuned weights are simply discarded and the pretraining starts from the initial weights of the adaptation process. Adam optimizer with same configuration is used to train this model.

FTML: The FTML updates the meta-parameters using MAML style update. For that, we trained the FTML with 5 inner-loop updates (each inner-loop have task-specific data for the inner-loop adaptation). After the meta-update, the meta-parameters are used as the initialization for the inner-loop adaptation, and similar to FTL the adapted weights are simply discarded after evaluation. The inner-loop is trained with SGD with a learning rate of 0.001 and the outer-loop is trained with a learning rate of 0.0005.

FOML: Our method also have two sets of parameter vectors, thus two different optimizers. We use Adam for both online and meta updates with a learning rate of 0.001 for both cases. Since, our online updates share the meta-knowledge via the regularization term, the β_1 controls how much pull is applied to the meta parameters by the online parameters. We use 0.01 for β_1. Similarly the pull on meta-parameters by the online parameters is controlled by β_2, and we use 0.001 in our experiments.