Supplementary Materials for Incremental Object Learning from Contiguous Views

Stefan Stojanov[1], Samarth Mishra[1], Ngoc Anh Thai[1], Nikhil Dhanda[1], Ahmad Humayun[1],
Chen Yu[2], Linda B. Smith[2], James M. Rehg[1]
Georgia Institute of Technology[1]
Indiana University Bloomington[2]
{sstojanov,smishra,athai6,nn3,ahmadh,rehg}@gatech.edu
{chenyu,smith4}@indiana.edu

1. Additional Details on Data Generation

1.1. Training Data

Lighting: Light sources available in CRIB are three rod shaped light sources and four point light sources. These are implemented using Blender's built in Lamp object, using type Point for point sources and Area for rod sources. After the choice of rod or point is made, the specified location and rotation (if rod light source) are applied. The color temperature of the light is implemented by using Blender's Blackbody shader and is applied.

Object motion: The user specifies the total number of frames, n (length of the sequence) and number of points on the viewing sphere p. A set of p key frames of specified or random (azimuth, elevation, tilt, scale) are inserted at random points between 0 and n. Note that scale here is change from the object's initial scale. Once these keyframes are specified, linear interpolation is done using Blender's animation tools to generate the object motion, rotating around its estimated center of mass.

Preprocessing: For the collected Toys-200 automatically preprocessing the geometry is straightforward since their geometry was manually cleaned up and joined when they were collected. Importing ShapeNet objects in Blender objects results in importing all the edges and faces as individual objects. We implement code that can appropriately join this geometry into a single object. For all objects the center of mass is estimated using built in tools based on the shape of the object. Since the object is rotating and changing in scale, we need to make sure that during the motion it will not leave the camera field of view. This is done by calculating the initial scale of the object so that the 3D bounding box during the interpolated change in rotation and scale will not be greater than the maximum 3D bounding box that fits in the camera field of view.

Annotations: Instance segmentations are generated by rendering an alpha mask of the object. This results in an image where pixels where the object is present are 255 and pixels where the object is absent are 0. The segmentation is used to calculate an bounding box region for the object.

Backgrounds: Backgrounds and foregrounds (objects) are rendered separately. When compositing an object sequence with a background we make sure that the object is not included in the background set of objects. For Toys-200 there are 8 background scenes consisting of groups of 25 objects rendered laid out on a floor in a cluttered manner. For ShapeNet, there are 4 background scenes each consisting of 20 objects coming from 5 categories. We ensure that the objects in the background are distinct by choosing backgrounds from scenes where the object is absent. Compositing is done using a standard alpha overlay.

Video: Please refer to our attached video for examples of the CRIB data generation process for training and testing data for both Toys-200 and ShapeNet.

2. Description of Algorithms

Since our task is different from [1, 5, 3] at most one new concept is present in each learning exposure, we reimplement the algorithms in PyTorch [4], with modifications as described in the following sections. Refer to §4 for details of additional data augmentation performed on each dataset. In all of the algorithms, the images are input to the network after the following normalization: $\text{image} = \frac{\text{image}-\text{mean\_image}}{255}$, where the operations are pixel-wise subtraction and division, and mean_image is the pixel-wise mean over all images in the training data seen by the algorithm.

2.1. LwF and iCaRL

At each time step, training data for the new concept includes positive bounding box patches from each frame in a data unit and as many random patches (one per data unit frame) outside of the positive patch as negative samples to allow for discriminative learning. In addition to samples from the new concept, iCaRL also includes images for the learned concepts from the exemplar sets.

For iCaRL, after the model finishes updating the feature representations for the new concept, exemplar set for that concept is constructed, consisting of the positive patches of the most representative images of that concept.

The algorithm performance is tested after each learning exposure on positive patches of the testing images as described in the main document.

2.2. E2EIL

As described in [1], E2EIL has an initial training phase followed by a balanced finetuning phase at each time step. In the initial training phase, training data comes from positive patches of the new data unit and the exemplar sets for the previously learned concepts. Note that negative samples as described in §2.1, are only used in the very first learning exposure. An initial exemplar set is then constructed as described in [1], after which, balanced finetuning is done followed by updating the exemplar set with the final most representative images. Testing is performed as described in §2.1.

3. Hyperparameter Settings

3.1. For experiments on CRIB-Toys

The hyperparameters presented in Tables 1 and 2 are used for training the deep network associated with a learning algorithm on each exposure to a new data unit. We arrived at these hyperparameters via grid search over initial learning rate, learning rate decay and total number of epochs. The training procedure uses a Stochastic Gradient Descent Optimizer with momentum of 0.9 and a weight decay (coefficient of L2 penalty term over parameters) of 10^{-5}. For LwF and iCaRL, each of these training sessions starts off with an initial learning rate (Init LR train) and follows a decay schedule of being reduced by a factor of Learning Rate Decay (LRD) at 0.7 and 0.9 mark of the total number of epochs (Num. epochs train). For E2EIL, the learning rate (for both the initial training–LR train and balanced finetuning–LR finetune) is decayed every 10 epochs.

3.2. For experiments on CRIB-ShapeNet

The hyper-parameters and the training procedure used for iCaRL-PT-ND on CRIB-ShapeNet are exactly the same as those used for it in the experiment on CRIB-Toys (found in Table 1.

3.3. For experiments on CIFAR

3.3.1 Single Exposures on CIFAR-100

On CIFAR,-100 we use an initial learning rate of 0.05 for iCaRL-PT-ND and 0.2 for iCaRL-S-ND. The model is trained for 70 epochs. The training procedure uses a Stochastic Gradient Descent Optimizer with momentum of 0.9 and a weight decay of 10^{-5}. Learning rate is reduced by a factor of 5 at 0.7 and 0.9 mark of the total number of epochs.

3.3.2 Repeated Exposures on CIFAR-20

On CIFAR-20 (20 category subset of CIFAR-100) we use an initial learning rate of 0.002 for iCaRL-PT-ND. The model is trained for 5 epochs. The training procedure uses a Stochastic Gradient Descent Optimizer with momentum of 0.9 and a weight decay of 10^{-5}. Learning rate is reduced by a factor of 5 at 0.7 and 0.9 mark of the total number of epochs.

4. Data Augmentation

This section describes data augmentations used to aid training.

4.1. For experiments on CRIB-Toys

4.1.1 LwF and iCaRL

We use random crops by zero-padding on all sides and cropping out a random section of the same size as the original image from this padded image. Random flip augmentations are also used where flipping is done across the vertical axis (i.e. the image is flipped horizontally). In addition, we use color augmentation by jittering the hue, saturation and value of each image by random values in the range $[-0.02, 0.02]$, $[-0.05, 0.05]$ and $[-0.1, 0.1]$ respectively.

4.1.2 E2EIL

We choose randomly with equal probability, one of the following three operations to perform on top of the raw image:

1. Brightness augmentation: Intensity is altered by adding a random integer value in the range $[-63, 63]$
2. Multiplicative jittering: Each pixel in the normalized image is multiplied by a random value in the range $[0.2, 1.8]$.
3. Keeping the original image unaltered

Then we randomly (with probability $1/2$) decide whether to crop the image. Random cropping is performed as described in §4.1.1. Finally, we randomly (with probability $1/2$) decide whether to horizontally mirror the image.

Table 1. Hyperparameter Settings for Fully Supervised Experiments on CRIB-Toys

	Hyperparameters	LwF	iCaRL-S-(D/ND)	iCaRL-PT-ND	E2EIL-S-(D/ND)	E2EIL-PT-(D/ND)
Single Exposure CRIB-Toys-200 & Repeated Exposure CRIB-Toys-(50/200)	Init LR train	0.01	0.01	0.01	0.01	0.01
	Init LR finetune	-	-	-	0.001	0.001
	LRD	5	5	5	10	10
	Num. epochs train	20	10	20	15	20
	Num. epochs finetune	-	-	-	10	14

Table 2. Hyperparameter Settings for Unsupervised Repeated Exposure Experiments on CRIB-Toys

	Hyperparameters	iCaRL-PT-ND
Repeated Exposure CRIB-Toys-(50/100/150/200)	Init LR train	0.01
	LRD	5
	Num. epochs train	10

4.2. For experiments on CRIB-ShapeNet

The data augmentations used for iCaRL-PT-ND on CRIB-ShapeNet are the ones as described in §4.1.1.

4.3. For experiments on CIFAR

For the single exposure experiment, we perform random crops and random flip augmentations as described in §4.1.1. For the repeated exposure experiment, color augmentation as described in §4.1.1 is also applied in addition to the operations performed in single exposure.

5. Modifications to Loss Function

5.1. LwF

We modify the loss to accommodate for our setting of learning at most one new concept in each learning exposure. We use binary cross-entropy loss instead of multi-class cross-entropy loss for classification and distillation. Hence we try to minimize the following:

$$L(w) = L_D(w) + L_C(w)$$

where $L_D(w)$ is defined as

$$L_D(w) = \sum_{i=1}^{N} \sum_{j=1}^{C-1} [\, pdist_{ij} \log q_{ij} + (1 - pdist_{ij}) \log\, (1 - q_{ij})\,]$$

and $L_C(w)$ is defined as

$$L_C(w) = \sum_{i=1}^{N} (p_{ij} \log q_{ij} + (1 - p_{ij}) \log\, (1 - q_{ij}))$$

where p_i is the ground truth label for sample i and q_i is the score obtained by applying a sigmoid function on the logits from the network, with parameters w. $pdist_i$ is the output of the previous network (before any updates are made to w in the current time step) for sample i. N is the number of samples in a minibatch and C is the number of unique concepts seen by (hence, also the number of output units in) the learning algorithm.

5.2. E2EIL

The loss is composed of classification loss and distillation loss. Classification loss is a multi-class cross entropy loss, using the ground truth as the target label for all images and is computed across all learned concepts. Distillation loss is computed in a similar manner to the distillation loss in [3] with a temperature of 2. We use an additional coefficient for the distillation loss as suggested in [2], in which distillation loss has a weight of T^2. We minimize the following loss function:

$$L(w) = L_D(w) \cdot T^2 + L_C(w)$$

The classification loss $L_C(w)$ is defined as:

$$L_C(w) = \frac{-1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} p_{ij} \log q_{ij}$$

where p_i is the ground truth label for sample i and q_i is the score obtained by applying a softmax function on the logits from the network.
The distillation loss $L_D(w)$ is defined as:

$$L_D(w) = \frac{-1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} pdist_{ij} \log qdist_{ij}$$

where $pdist_i$ and $qdist_i$ are modified versions of p_i and q_i respectively, as described in §3.2 of [1].

6. Modifications in iCaRL and E2EIL for repeated exposures

Modifications are required in order to allow our adaptation of iCaRL [5] and E2EIL [1] to work with repeated exposure to concepts seen in previous learning exposures. The original algorithms operate under the assumption that every new learning exposure contains a new concept, and make architectural changes (adding a new output node) and construct the exemplar set. We encourage the reader to refer to Algorithm 4 in [5] for specific details of the standard iCaRL exemplar set construction method.

Our modifications are so that when iCaRL (or E2EIL) receives training examples for a concept seen before, exemplars previously stored for that concept are combined with the new training images. Assume that iCaRL (or E2EIL) is undergoing a repeated exposure for concept k. It might be the case that the existing exemplar set E_k for concept k is so that $|E_k| < |D_k|$ where D_k is the set of current training images for k. In order to avoid over-representation of D_k over E_k when constructing the exemplar set, we use a weighted mean where the weights for E_k images are proportional to $|D_k|$ and vice versa.

7. Inference Procedure for Unsupervised Repeated Exposure

In this section, we define the inference procedure (and accuracy computation) for a learning algorithm in the unsupervised repeated exposure learning task.

In this case a learning algorithm generates an internal set of labels for previously seen objects. The generated labels are ascertained by solving a maximum weight bipartite matching problem.

7.1. Testing Accuracy

Calculating testing accuracy is solved at inference time using testing samples from objects that the learning algorithm has already been exposed to. Define $\mathcal{A}$ and $\mathcal{B}$ as the set of ground truth labels and the set of labels provided by the learning algorithm respectively until a certain learning exposure. The nodes in this matching problem are the ground truth classes (from $\mathcal{A}$) on one side and the labels provided by the learning algorithm (from $\mathcal{B}$) on the other. The weight of an edge $A - B$ is the number of times during testing, that an object with a ground truth label $A \in \mathcal{A}$ was classified as a class $B \in \mathcal{B}$ by the learning algorithm. Once this matching has been done, the accuracy for each ground truth class is computed using this matching. If a ground truth class doesn't get matched to any class in $\mathcal{B}$, it gets an accuracy of 0.

References

[1] F. M. Castro, M. J. Marin-Jimenez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 233–248, 2018. 1, 2, 3, 4

[2] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[3] Z. Li and D. Hoiem. Learning without Forgetting. In *European Conference on Computer Vision (ECCV)*, pages 614–629, 2016. 1, 3

[4] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017. 1

[5] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. iCaRL: Incremental Classifier and Representation Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 4