A. Proof of Theorem 1

Theorem 1. *Given a discriminative model $f_\theta(x)$, the unbiased gradient estimator of corresponding Energy-based model $\log p_\theta(x)$ is given by*

$$\mathbb{E}_{p_\theta(y|x)}[y^T \nabla_\theta f_\theta(x)] - \mathbb{E}_{p_\theta(x,y)}[y^T \nabla_\theta f_\theta(x)].$$

Proof. Notice that we could derive ELBO of $\log p_\theta(x)$ as:

$$\log p_\theta(x) \geq \mathbb{E}_{q(y|x)}\big[\log \frac{p_\theta(x,y)}{q(y|x)}\big], \tag{8}$$

and we know the maximal would be obtained when $KL(q(y|x)\|p_\theta(y|x)) = 0$, which implies that optimal $q^*(y|x)$ is $p_\theta(y|x)$. Thus, we will have

$$\begin{aligned}
\log p_\theta(x) &= \mathbb{E}_{p_\theta(y|x)}\big[\log \frac{p_\theta(x,y)}{p_\theta(y|x)}\big] &(9)\\
&= \mathbb{E}_{p_\theta(y|x)}\big[\log p_\theta(x,y) - \log p_\theta(y|x)\big] &(10)\\
&= \mathbb{E}_{p_\theta(y|x)}\big[\log \frac{\exp(y^T f_\theta(x))}{\int_x \sum_y \exp(y^T f_\theta(x))} - \log \frac{\exp(y^T f_\theta(x))}{\sum_y \exp(y^T f_\theta(x))}\big] &(11)\\
&= \mathbb{E}_{p_\theta(y|x)}\big[\log \sum_y \exp(y^T f_\theta(x)) - \log \int_x \sum_y \exp(y^T f_\theta(x))\big]. &(12)
\end{aligned}$$

Thus, we could obtain $\nabla_\theta \log p_\theta(x)$ by taking derivative of eq (12):

$$\begin{aligned}
&\nabla_\theta \log p_\theta(x) &(13)\\
&= \mathbb{E}_{p_\theta(y|x)}\big[\nabla_\theta\big[\log \sum_y \exp(y^T f_\theta(x))\big] - \nabla_\theta\big[\log \int_x \sum_y \exp(y^T f_\theta(x))\big]\big] &(14)\\
&= \mathbb{E}_{p_\theta(y|x)}\big[\frac{\nabla_\theta(\sum_y \exp(y^T f_\theta(x))}{\sum_y \exp(y^T f_\theta(x))} - \frac{\nabla_\theta(\int_x \sum_y \exp(y^T f_\theta(x))}{\int_x \sum_y \exp(y^T f_\theta(x))}\big] &(15)\\
&= \mathbb{E}_{p_\theta(y|x)}\big[\frac{\sum_y \nabla_\theta(\exp(y^T f_\theta(x))}{\sum_y \exp(y^T f_\theta(x))} - \frac{\int_x \sum_y \nabla_\theta(\exp(y^T f_\theta(x))}{\int_x \sum_y \exp(y^T f_\theta(x))}\big] &(16)\\
&= \mathbb{E}_{p_\theta(y|x)}\big[\frac{\sum_y \exp(y^T f_\theta(x)\nabla_\theta(\log\exp(y^T f_\theta(x))}{\sum_y \exp(y^T f_\theta(x))} - \frac{\int_x \sum_y \nabla_\theta(\exp(y^T f_\theta(x))}{\int_x \sum_y \exp(y^T f_\theta(x))}\big] &(17)\\
&= \mathbb{E}_{p_\theta(y|x)}\big[\sum_y p_\theta(y|x) y^T \nabla_\theta f_\theta(x) - \frac{\int_x \sum_y \exp(y^T f_\theta(x)\nabla_\theta(\log\exp(y^T f_\theta(x))}{\int_x \sum_y \exp(y^T f_\theta(x))}\big] &(18)\\
&= \mathbb{E}_{p_\theta(y|x)}\big[\sum_y p_\theta(y|x) y^T \nabla_\theta f_\theta(x) - \int_x \sum_y p_\theta(x,y) y^T \nabla_\theta f_\theta(x)\big] &(19)\\
&= \mathbb{E}_{p_\theta(y|x)}\big[\mathbb{E}_{p_\theta(y|x)}[y^T \nabla_\theta f_\theta(x)] - \mathbb{E}_{p_\theta(x,y)}[y^T \nabla_\theta f_\theta(x)]\big] &(20)\\
&= \mathbb{E}_{p_\theta(y|x)}[y^T \nabla_\theta f_\theta(x)] - \mathbb{E}_{p_\theta(x,y)}[y^T \nabla_\theta f_\theta(x)]. &(21)
\end{aligned}$$

Notice that the outer expectation could be taken off since after inner expectation, there won't be any randomness on y. □

B. Proof of Theorem 2

Theorem 3 The estimation of gradient of loss used in training the proposed method eq (7) is given by

$$\begin{aligned}&\nabla_\theta L(\theta; p_\theta) \triangleq \nabla_\theta KL(q_t(\theta|D_{1:t}) \| q_{t-1}(\theta|D_{1:t-1}))\\&- \nabla_\theta \log p_\theta(y|x) + \gamma(\mathbb{E}_{p_\theta(x,y)}\left[y^T \nabla_\theta f_\theta(x)\right] - \mathbb{E}_{D_t}[y^T \nabla_\theta f_\theta(x)]).\end{aligned}$$

Proof. To solve the objetive function:

$$\min_{q_t \in \mathbf{Q}} \mathbb{E}_{q_t, D_t}\left[- (1-\lambda) \log p_\theta(y,x) - \lambda \log p_\theta(y|x) - \lambda \log p_\theta(x) + KL\left[q_t(\theta|D_{1:t}) \| q_{t-1}(\theta|D_{1:t-1})\right]\right],$$

we need to take derivative over the above equation and calculate the unbiased gradient estimator of each term. Notice that the first term is provided in (6) and the second term is provided in Theorem 1. Thus, we could substitute these values into the equation to get:

$$\begin{aligned}\nabla_\theta L(\theta; p_\theta) \triangleq& -(1-\lambda)\nabla_\theta \log p_\theta(y,x) - \lambda \nabla_\theta \log p_\theta(y|x) - \lambda \nabla_\theta \log p_\theta(x) +\\&\nabla_\theta KL(q_t(\theta|D_{1:t}) \| q_{t-1}(\theta|D_{1:t-1}))\\=&(1-\lambda)(\mathbb{E}_{p_\theta(x,y)}\left[y^T \nabla_\theta f_\theta(x)\right] - \mathbb{E}_D\left[y^T \nabla_\theta f_\theta(x)\right])\\&- \lambda \nabla_\theta \log p_\theta(y|x) + \nabla_\theta KL(q_t(\theta|D_{1:t}) \| q_{t-1}(\theta|D_{1:t-1}))\\&+ \lambda \mathbb{E}_{p_\theta(x,y)}[y^T \nabla_\theta f_\theta(x)]) - \lambda(\mathbb{E}_{p_\theta(y|x)}[y^T \nabla_\theta f_\theta(x)]\\=&\nabla_\theta KL(q_t(\theta|D_{1:t}) \| q_{t-1}(\theta|D_{1:t-1})) - \lambda \nabla_\theta \log p_\theta(y|x) + \mathbb{E}_{p_\theta(x,y)}\left[y^T \nabla_\theta f_\theta(x)\right]\\&- \mathbb{E}_{x_b \sim D_t} \mathbb{E}_{y_b \sim \lambda p_\theta(y|x_b) + (1-\lambda) D_t}[{y_b}^T \nabla_\theta f_\theta(x_b)],\end{aligned}$$

where x_b is the training instance sampled from true data distribution D_t with y_b sampled from a mixture of conditional $p_\theta(y|x_b)$ and training sets. Again, to generate the samples (x_t, y_t) from the current model, we exploit the hybrid Monte-Carlo (Neal et al.), specifically the Langevin dynamics sampler, as listed in Algorithm 1.

In this work, we treat the generative term as a regularization to alleviate catastrophic forgetting in discriminative task. Therefore, we relax the constant λ and introduce a new hyperparameter γ to represent the importance of the generative regularization. For simplicity, we only draw samples from true data distribution instead of a mixture of conditional $p_\theta(y|x_b)$ and training sets. This leads to the final estimation of gradient of loss used in training the proposed method:

$$\begin{aligned}&\nabla_\theta L(\theta; p_\theta) \triangleq \nabla_\theta KL(q_t(\theta|D_{1:t}) \| q_{t-1}(\theta|D_{1:t-1}))\\&- \nabla_\theta \log p_\theta(y|x) + \gamma(\mathbb{E}_{p_\theta(x,y)}\left[y^T \nabla_\theta f_\theta(x)\right] - \mathbb{E}_{D_t}[y^T \nabla_\theta f_\theta(x)]).\end{aligned}$$

□

C. Examples of Generated Images

We show samples of the generated MNIST digit in Figure 9 and samples of the generated Fashion-MNIST in Fiagure 10. These images are generated by using Multi-layer Perceptron model (MLP) with 2 hidden layers and each layer has dimension 256 with ReLU activation function. Notice that our main task is to overcome catastrophic forgetting but not image generation. Generative capability is just used as a regularization term so the images generated are not perfectly following the true data distribution. In addition, we are using a rather small model to build EBM. In practice, people reported to use much larger networks (about 20 times more parmeters) in order to generate more clear images for CIFAR-10 dataset (Du & Mordatch, 2019).

D. Preprocess of Data

For all the dataset, we normalized the pixel values in range [0,1]. For MNIST and Fashion-MNIST , we have the train, validation and test splits provided within the dataset. Images in CUB dataset is rather limited. Most classes have samples

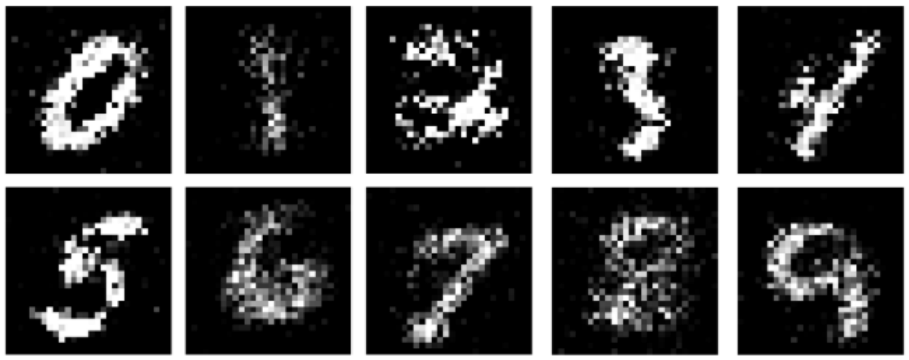

Figure 9. Examples of generated MNIST images. The first row shows digits 0 to 4 and the second row shows digits 5 to 9.

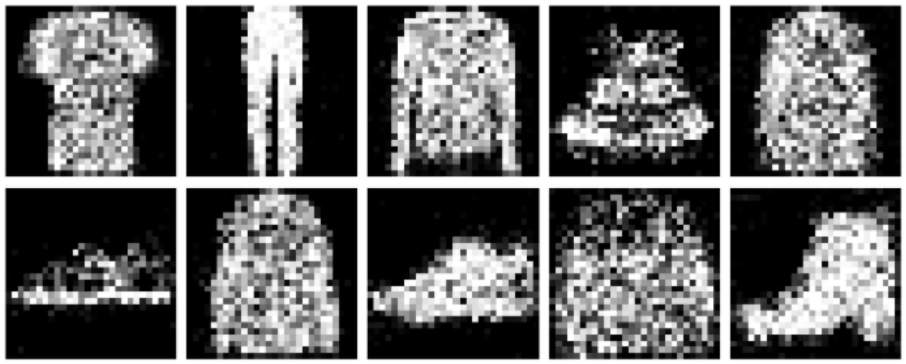

Figure 10. Examples of generated Fashion-MNIST images. The first row shows digits 0 to 4 and the second row shows digits 5 to 9. The first row corresponds to objects T-shirt, Trouser, Pullover, Dress and Coat. The second row corresponds to Sandal, Shirt, Sneaker, Bag and Ankle boot.

less than 100. Therefore, we select only the top 100 classes with more images and randomly pick 40 to form the train set and 10 to be validation set. The rest (non-fixed number) of the remaining images will be left as test set. In addition, CUB dataset provided foreground and background segmentation. We segment only the foreground bird images and left the background to be black. Without this, EBM will try to generate background istead and this will not benefit to overcoming forgetting.

E. Implementation Details

For each dataset/task, we compare these methods under the same network architecture. As illustrated in the related work, we basically do not compare memory-based and model adaption methods as the data and model complexity will increase, but we include generative-based model VGR to compare. The implementation of VGR is primarily based on these two repositories with adjustment toward our setup[3]. For EWC and VCL, we follow the released open source implementation [4]. The chosen baseline methods represent the state-of-the-art algorithms to overcome forgetting without changing model or adding data. For Permuted-MNIST and Split-MNIST, we use a Multi-layer Perceptron model (MLP) with 2 hidden layers and each layer has dimension 256. ReLU is used as the activation function. For Permuted-MNIST, we use single-head model and for Split-MNIST we use multi-head model. For Fashion-MNIST dataset, we evaluate the results on Convolutaional Neural Networks (CNN) with 4 layers of convolutional layer (32,1), (64,32), (64,64), (64,64) followed by one layer of fully connected layer. For CUB dataset, we apply a Wide-Residual Network (Zagoruyko & Komodakis, 2016) implemented with depth 16 and widen-factor 2. The implementation could be found on the official Pytorch repository[5]. All the models are trained with an ADAM optimizer.

[3]https://github.com/nbro/Continual-learning-1,https://github.com/GMvandeVen/continual-learning

[4]https://github.com/nvcuong/variational-continual-learning

[5]https://github.com/meliketoy/wide-resnet.pytorch

The hyperparameters used in the experiment are listed in the Table 3. In addition, after each step of SGLD sampling, we will clamp the sample within range [0,1] to make sure the generated image is within the range of true data distribution. Learning rate, Adam Beta, SGLD step size, SGLD noise follows previous implementation of SGLD sampling [6] or WRN model [7]. Number of models sampled from the Bayesian posterior is mostly limited by time constraints. In general we found out 3 is enough but the more the better. For Epochs of each round of SGLD update, more steps is better, but more updates will also lead to very long training time. Thus, in practice we try some numbers from 10 to 100 steps on small portion of data. We will stop searching bigger numbers once the model could generate images look similar to data distribution. Buffer reinitialization rate is determined from validation set. We search over .05, .2 and .5.

	Permuted	Split	Fashion	CUB
Learning Rate	1e-3	1e-3	1e-3	1e-4
Adam Beta	(0,0.999)	(0,0.999)	(0,0.999)	(.9, .999)
Number of models sampled from $p(\theta)$	10	10	10	3
Generation Importance γ	1	1	1	.2
Buffer Size	10000	10000	10000	200
SGLD step size	10	10	10	1
Buffer Reinitialization Rate	.05	.5	.05	.05
SGLD noise	5e-3	5e-3	5e-3	1e-2
Epochs of each round of SGLD update	60	60	5	20

Table 3. Summarization of hyperparameters used in each task.

[6] https://github.com/rosinality/igebm-pytorch
[7] https://github.com/kibok90/iccv2019-inc