BooVAE: A Scalable Framework for Continual VAE Learning under Boosting Approach

Anonymous Authors
Anonymous Institution

1. Introduction

Since most of the real-world datasets are unlabeled, unsupervised learning is an essential part of the machine learning field. Generative models allow us to obtain samples from observed empirical distributions of the complicated high-dimensional objects such as images, sounds or texts. This work is mostly devoted to VAEs with the focus on incremental learning setting.

It was observed that VAEs ignore dimensions of latent variables and produce blurred reconstructions (Burda et al., 2015; Sønderby et al., 2016). There are several approaches to address these issues, including amortization gap reduction (Kim et al., 2018), KL-term annealing (Cremer et al., 2018) and alternative optimization objectives introduction (Rezende and Viola, 2018). In all cases, it was observed that the choice of the prior distribution is highly important and use of default Gaussian prior overregularizes the encoder.

In this work, we address the problem of constructing the optimal prior for VAE. The form of optimal prior (Tomczak and Welling, 2018) was obtained by maximizing a lower bound of the marginal likelihood (ELBO) as the aggregated posterior over the whole training dataset. To construct reasonable approximation, we consider a method of greedy KL-divergence projection. Applying the maximum entropy approach allows us to formulate a feasible optimization problem and avoid overfitting. The greedy manner in which components are added to prior reveals a high potential of the method in an incremental learning setting since we expect prior components to store information about previously learned tasks and overcome catastrophic forgetting (McCloskey and Cohen, 1989; Goodfellow et al., 2013; Nguyen et al., 2017).

From practitioners point of view, it is essential to be able to store one model, capable of solving several tasks arriving sequentially. Hence, we propose the algorithm with one pair of encoder-decoder and update only the prior. We validate our method on the disjoint sequential image generation tasks. We consider MNIST and Fashion-MNIST datasets.

2. Background

2.1. Variational Autoencoders

VAEs consider two-step generative process by a prior over latent space $p(z)$ and a conditional generative distribution $p_\theta(x|z)$, parametrized by a deep neural network. Given empirical data distribution $p_e(x) = \frac{1}{N}\sum_{n=1}^{N}\delta(x - x_n)$ we aim at maximizing expected marginal log-likelihood. Following the variational auto-encoder architecture, amortized inference is proposed by choosing variational posterior distribution $q_\phi(z|x)$ to be parametrized by DNN, resulting in a following objective:

$$\max_{\phi,\theta} \mathbb{E}_x \mathcal{L}(x, \phi, \theta) = \max_{\phi,\theta} \mathbb{E}_x \mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - \mathbb{E}_x D_{\mathrm{KL}}[q_\phi(z|x) \| p(z)]. \tag{1}$$

To obtain a richer prior distribution one may combine variational and empirical bayes approaches and optimize the objective (1) over the prior distribution. For a given empirical density, optimal prior is a mixture of posterior distributions in all points of the training dataset. Clearly, such prior leads to overfitting. Hence, keeping the same functional form, truncated version with K presudoinputs was proposed (Tomczak and Welling, 2018) as VampPrior.

In the present paper, we address two crucial drawbacks of the VampPrior. Firstly, for large values of K variational inference will be very computationally expensive. Even for the MNIST dataset Tomczak and Welling (2018) used the mixture of 500 components. Secondly, it is not clear how to choose K for the particular dataset, as we have a trade-off between prior capacity and overfitting. For this purpose, we adapt maxentropy variational inference framework (Egorov et al., 2019). We add components to the prior during training in a greedy manner and show that in such setting, fewer components are needed for the comparable performance.

2.2. MaxEntropy Variational Inference

Assume, that we want to approximate complex distribution p^* by a mixture of simple components p_i Each component of the mixture is learned greedily. At the first stage we initialize mixture by a standard normal distribution. Afterwards, we add new component h from some family of distributions Q, with the weight α one by one $p_t = \alpha h + (1-\alpha)p_{t-1}$, $\alpha \in (0, 1)$ in two stages:

1. Find optimal $h \in Q$. We apply Maximum Entropy approach to minimize KL-divergence between mixture and target distribution. This task can be reformulated as a following KL-minimization problem (see Appendix A): $h = \arg\min_{h \in Q} D_{\mathrm{KL}}\left(h || \left[\frac{p^*}{p_{t-1}}\right]^\lambda\right)$.
2. Choose α, corresponding to the optimal h: $\alpha^* = \arg\min_{\alpha\in(0,1)} D_{\mathrm{KL}}(\alpha h + (1-\alpha)p_{t-1} || p^*)$.

3. Algorithm

In this work, we suggest combining boosting algorithm for density distributions with the idea of VampPrior to deal with the problem of catastrophic forgetting in the incremental learning setting.

Proposed algorithm for training VAE consists of two steps. On the first one we optimize evidence lower bound (1) w.r.t. the parameters of the encoder and decoder. On the second stage, we learn new component h for the prior distribution, and its weight α, keeping parameters of the encoder and decoder fixed. We learn each component to be posterior given the learnable artificial input u: $q_\phi(z|u)$ with target density being mixture of posteriors in all points from random subset $\mathcal{M}$ of the whoel training dataset $\mathcal{D}$. Parameters of the first component u_0 are obtained by ELBO maximization simultaneously with the network parameters, as shown in the Algorithm 1.

3.1. Incremental learning

In the incremental learning setting, we do not have access to the whole dataset. Instead, subsets $\mathcal{D}_1, \ldots \mathcal{D}_T$ arrive sequentially and may come from different domains. With the first task $\mathcal{D}_1$ we

Algorithm 1: BooVAE algorithm

```
Input: Dataset $\mathcal{D}: \{(x_i)\}_{i=1}^N$, $\lambda$, Maximal number of components $K$
Output: $p_K$, $\theta^*$, $\phi^*$
Choose random subset $\mathcal{M} \subset \mathcal{D}$ and initialize prior $p_0 = q_\phi(z|u_0)$ and k = 1;
$\theta^*, \phi^*, u_0 = \arg\max \mathcal{L}(p_0, \theta, \phi)$;
while not converged do
    Update network parameters $\theta^*, \phi^* = \arg\max \mathcal{L}(p_{k-1}, \theta, \phi)$;
    if $k \le K$ then
        Add new component    $p_k = \alpha^* h^* + (1 - \alpha^*) p_{k-1}$ by MaxEntropy VI ;
        $k = k + 1$;
    end
end
```

follow Algorithm 1 to obtain prior $p^{(1)}$ and optimal values of the network parameters. Starting from $t > 1$, we add regularization to ELBO, which ensures that the model keeps encoding and decoding learned prior components in the same manner (see Appendix B). Since we do not have access to the whole dataset anymore, the form of optimal prior also changes. We use prior of the previous step as a proxy for an optimal one (see Appendix C) $p^{*(t)} \approx \left(\frac{t-1}{t} p^{(t-1)} + \frac{1}{nt} \sum_{x \in \mathcal{M}_t} q_\phi(z|x)\right)$

4. Experiments

We perform experiments on MNIST dataset (LeCun et al., 2010), containing ten hand-written digits and on fashion MNIST (Xiao et al., 2017), which also has ten classes of different clothing items. Therefore, we have ten tasks in each dataset for sequential learning. To evaluate the performance of the VAE approach, we estimate a negative log-likelihood (NLL) on the test set, calculated by importance sampling method, with 5000 samples for each observation.

4.1. Offline setting

In an offline setting, we aim at comparing our method to VampPrior and Mixture of Gaussians prior. In the first case, each component is a posterior distribution given learnable pseudo-input, while in the second we learn each component as a Gaussian with diagonal covariance matrix in the latent space. For both priors, all the component are learned simultaneously. Table 1 clearly illustrates that BooVAE not only shows performance comparable to other data-driven priors but also requires fewer components.

	MNIST				Fashion MNIST			
# comp.	Std.	Vamp	MoG	Boo	Std.	Vamp	MoG	Boo
10		90.39 (0.13)	90.41 (0.25)	**89.98 (0.20)**		232.53 (0.06)	232.89 (0.12)	**231.94 (0.02)**
20		89.97 (0.11)	90.02 (0.08)	**89.78 (0.07)**		232.22 (0.07)	232.92 (0.12)	**231.84 (0.07)**
50	91.49 (0.06)	89.40 (0.09)	89.77 (0.13)	**89.16 (0.08)**	233.12 (0.02)	232.19 (0.08)	232.62 (0.23)	**231.63 (0.02)**
100		89.16 (0.17)	89.47 (0.12)	**88.90 (0.11)**		232.01 (0.03)	232.55 (0.22)	**231.55 (0.07)**
500		88.82 (0.09)	89.37 (0.03)	**88.68 (0.1)**		**231.67 (0.12)**	232.23 (0.26)	231.85 (0.13)

Table 1: NLL Results for different number of components in the prior. Offline setting

4.2. Incremental setting

In Table 2 we provide NLL results for incremental setting. The first column states how many tasks did the VAE see in total. We compare our approach with VAE with standard normal prior, Elastic weight consolidation (EWC) — model proposed by Kirkpatrick et al. (2017) to deal with catastrophic forgetting and model with MoG prior, where we add new components to the mixture for every task, just as we do in boosting, and apply the same regularization for the fair comparison.

	MNIST			Fashion MNIST		
# tasks	Std + EWC	MoG	Boo	Std + EWC	MoG	Boo
2	256.55 (8.38)	**96.50 (1.95)**	100.11 (1.39)	271.14 (6.05)	239.43 (2.76)	**227.83 (3.34)**
5	192.84 (0.12)	143.44 (8.05)	**132.08 (0.64)**	270.44 (0.98)	264.51 (1.93)	**253.12 (1.43)**
8	189.06 (4.72)	172.00 (12.93)	**140.80 (2.42)**	565.81 (22.94)	448.55 (103.92)	**260.05 (5.25)**
10	170.26 (2.20)	181.53 (29.02)	**142.92 (1.99)**	427.83 (21.19)	440.96 (49.75)	**284.86 (21.21)**

Table 2: NLL Results for incremental setting.

Results in the tables above demonstrate that BooVAE manages to overcome catastrophic forgetting better than pure EWC regularization with standard normal prior. Unstable NLL values for VAE with standard prior and EWC can be explained by the fact that some classes in the dataset are quite similar and even though model forgets old class, knowledge about a new class let her reconstruct it relatively good, resulting in deceptively better results. In Appendix D we provide results for each task separately to illustrate this effect further.

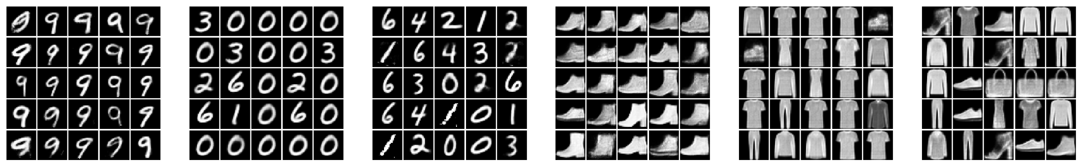

(*a*) Std + EWC (*b*) MoG (*c*) Boo (*d*) Std + EWC (*e*) MoG (*f*) Boo

Figure 1: Samples from prior after training on 10 tasks incrementally. Our approach is capable of sampling images from different tasks, while other methods either stick to the latest seen class, or are able to reproduce images from simplest classes

On Figure 1, we provide samples from the prior illustrating generation ability of all the methods after training on all of the ten tasks sequentially. Appendix E and F provide detailed qualitative and quantitative (Figure 3) evaluation of the samples diversity for different models.

5. Conclusion

In this work, we propose a method for learning a data-driven prior, using a MM algorithm which allows us to reduce the number of components in the prior distribution without the loss of performance. Based on this method, we suggest an efficient algorithm for incremental VAE learning which has single encoder-decoder pair for all the tasks and drastically reduces catastrophic forgetting.

References

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1086–1094, 2018.

Evgenii Egorov, Kirill Neklydov, Ruslan Kostoev, and Evgeny Burnaev. Maxentropy pursuit variational inference. In *International Symposium on Neural Networks*, pages 409–417. Springer, 2019.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pages 2683–2692, 2018.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2:18, 2010.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

Danilo Jimenez Rezende and Fabio Viola. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pages 3738–3746, 2016.

Jakub Tomczak and Max Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223, 2018.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Appendix A. MaxEntropy VI: objective derivation

$$\max_{h \in Q} \mathcal{H}(h)$$
$$D_{\mathrm{KL}}(p_{t-1}|p^*) - D_{\mathrm{KL}}(p_t|p^*) > 0$$

Expanding second term of the objective, we get:

$$\begin{aligned} D_{\mathrm{KL}}(p_t|p^*) = \int (p_{t-1} + \alpha(h - p_{t-1})) \left(\log \frac{p_{t-1}}{p^*} + \log \left(1 + \alpha \frac{h - p_{t-1}}{p_{t-1}} \right) \right) = \\ = D_{\mathrm{KL}}(p_{t-1}|p^*) + \alpha \int (h - p_{t-1}) \left(\log \frac{p_{t-1}}{p^*} + \log \left(1 + \alpha \frac{h - p_{t-1}}{p_{t-1}} \right) \right) + \\ + \int p_{t-1} \log \left(1 + \alpha \frac{h - p_{t-1}}{p_{t-1}} \right) \end{aligned}$$

Using first-order Taylor expansion we can simplify some terms:

$$\int p_{t-1} \log \left(1 + \alpha \frac{h - p_{t-1}}{p_{t-1}} \right) \approx \alpha \int p_{t-1} \frac{h - p_{t-1}}{p_{t-1}} = 0$$
$$\int (h - p_{t-1}) \log \left(1 + \alpha \frac{h - p_{t-1}}{p_{t-1}} \right) \approx \alpha \int \frac{(h - p_{t-1})^2}{p_{t-1}}$$

The constraint becomes of the following form:

$$D_{\mathrm{KL}}(p_{t-1}|p^*) - D_{\mathrm{KL}}(p_t|p^*) \approx -\alpha \int (h - p_{t-1}) \log \frac{p_{t-1}}{p^*} - \alpha^2 \int \frac{(h - p_{t-1})^2}{p_{t-1}}$$

Omitting terms, which do not depend on h and considering only first order terms result in optimization problem:

$$\max_{h \in Q} \mathcal{H}(h) + \lambda \int h \log \frac{p^*}{p_{t-1}} = \min_{h \in Q} D_{\mathrm{KL}} \left(h || \left[\frac{p^*}{p_{t-1}} \right]^{\lambda} \right)$$

Appendix B. ELBO for incremental setting

When we start training a new task, we already have a prior distribution of the following form:

$$p^{(t-1)} = \frac{1}{t-1} \sum_{j=1}^{t-1} \sum_{i=1}^{K} \alpha_{ij} \mathcal{N}(z|\mu_{ij}, \sigma_{ij}), \quad \text{where} \mathcal{N}(z|\mu_{ij}, \sigma_{ij}) := q_{\phi_j}(z|u_{ij}).$$

That is being said, we store prior components from the previous task in the form of Gaussian mean and variance vectors, computed using optimal encoder parameters for the given task (after the convergence).

To make sure that model keeps encoding and decoding prior components as it did during the training of the corresponding task, we add a regularization term to ELBO. For that purpose at the end of each task we compute reconstructions of the components' mean value $p_{\theta_j}(x|\mu_{i,j}) = p_{ij}(x)$.

$$\begin{aligned}
R^{ij}_{enc}(\phi) &= D_{\mathrm{KL}}\left[\mathcal{N}(z|\mu_{ij},\sigma_{ij})\|q_\phi(z|x_{ij})\right],\\
R^{ij}_{dec}(\theta) &= D_{\mathrm{KL}}\left[p_{ij}(x)\|p_\theta(x|z_{ij})\right], x_{ij}\sim p_{ij}(x),\ z_{ij}\sim\mathcal{N}(z|\mu_{ij},\sigma_{ij})\\
R_{tot}(\phi,\theta) &= \frac{1}{t-1}\sum_{j=1}^{t-1}\sum_{i=1}^{K}\alpha_{ij}\left(R^{ij}_{enc}(\phi)+R^{ij}_{dec}(\theta)\right)
\end{aligned}$$

All thing considered, objective for the maximization step is the following:

$$\max_{\phi,\theta}\mathbb{E}_{x\sim\mathcal{D}_t}\mathbb{E}_{z\sim q(z)}[\log p_\theta(x|z)] - \mathbb{E}_{x\sim\mathcal{D}_t}D_{\mathrm{KL}}[q_\phi(z|x)\|p(z)] - \varepsilon R_{tot}(\phi,\theta).$$

Appendix C. Optimal Prior in Incremental Setting

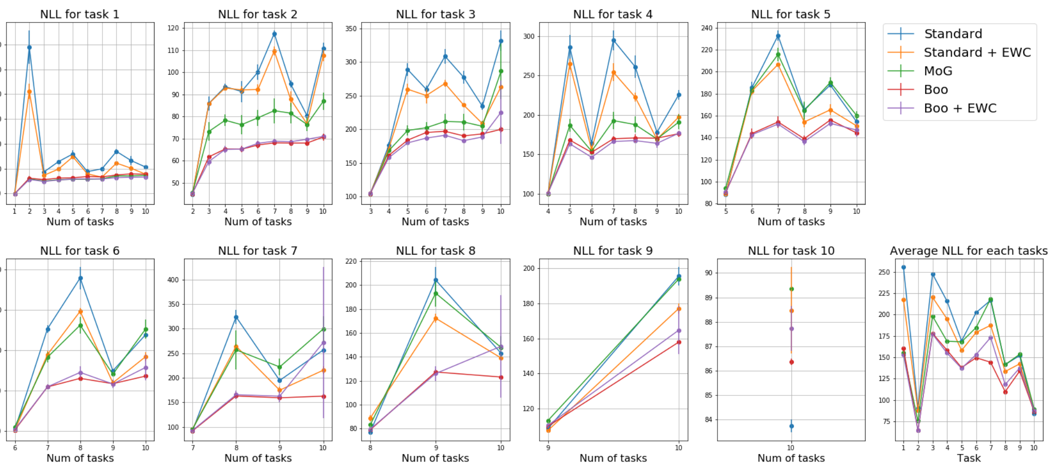
When training a new component and its' weight, we want optimal prior that we approximate to be close to the mixture of variational posteriors at all the training points, seen by the model. Since we don't have access to the data from the previous tasks, we suggest using trained prior as a proxy for the corresponding part of the mixture. Therefore, optimal prior for tasks $1:t$ can be expressed, using optimal prior from the previous step and training dataset from the current stage:

$$\begin{aligned}
p^{*(t)} =& \frac{1}{N_{1:t}}\sum_{x\in\mathcal{D}_{1:t}} q_\phi(z|x) = \frac{1}{N_{1:t}}\left(\sum_{x\in\mathcal{D}_{1:t-1}} q_\phi(z|x) + \sum_{x\in\mathcal{D}_t} q_\phi(z|x)\right) =\\
&\frac{1}{N_{1:t}}\left(N_{1:t-1}p^{*(t-1)} + \sum_{x\in\mathcal{D}_t} q_\phi(z|x)\right)
\end{aligned}$$

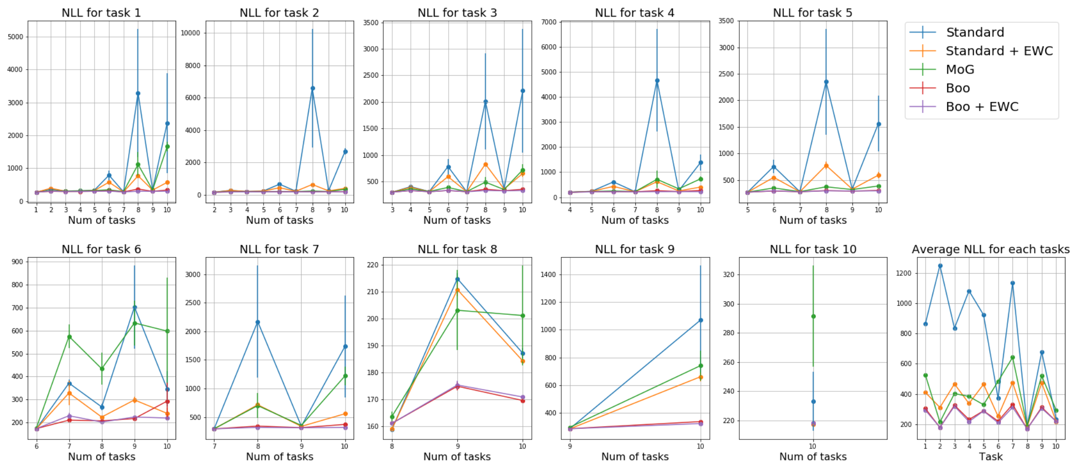
During training, we approximate optimal prior by the mixture $p^{(t)}$, using random subset containing n observations of the given dataset $\mathcal{M}_t \subset \mathcal{D}_t$. Therefore, the optimal prior can be approximated in the following way:

$$p^{*(t)} \approx \left(\frac{t-1}{t}p^{(t-1)} + \frac{1}{nt}\sum_{x\in\mathcal{M}_t} q_\phi(z|x)\right)$$

Appendix D. Per task performance on incremental setting



(*a*) MNIST



(*b*) fashion MNIST

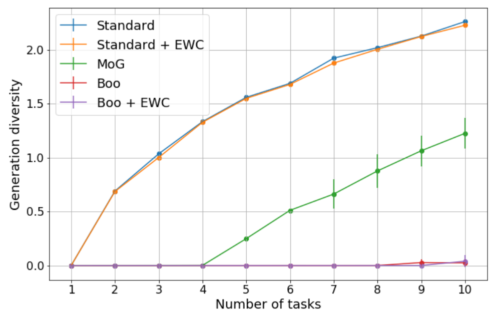
Figure 2: KL divergence between uniform and generated distribution.

Appendix E. Generation Diversity Evaluation

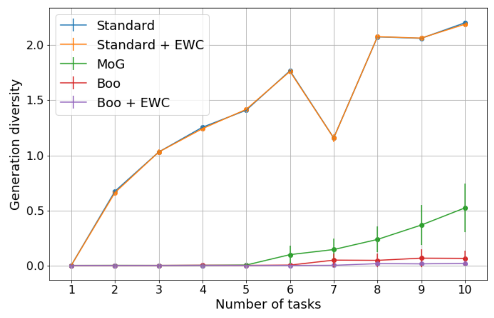
To evaluate diversity of the generated images, we calculate KL-divergence between Bernoulli distribution with equal probability for each class and empirical distribution of classes generated by the model.

$$\sum_k D_{\text{KL}}\left(u||\widehat{x}_k\right), \; u \sim \text{Be}\left(\frac{1}{K}\right), \; \widehat{x}_k \sim \text{Be}\left(\frac{N_k}{N}\right).$$

Since we want to assign classes to generated images automatically, we train classification network, which can classify images with high probability (more than 90%), use it to label generated objects and calculated the empirical distribution over 10000 generated samples.



(*a*) MNIST

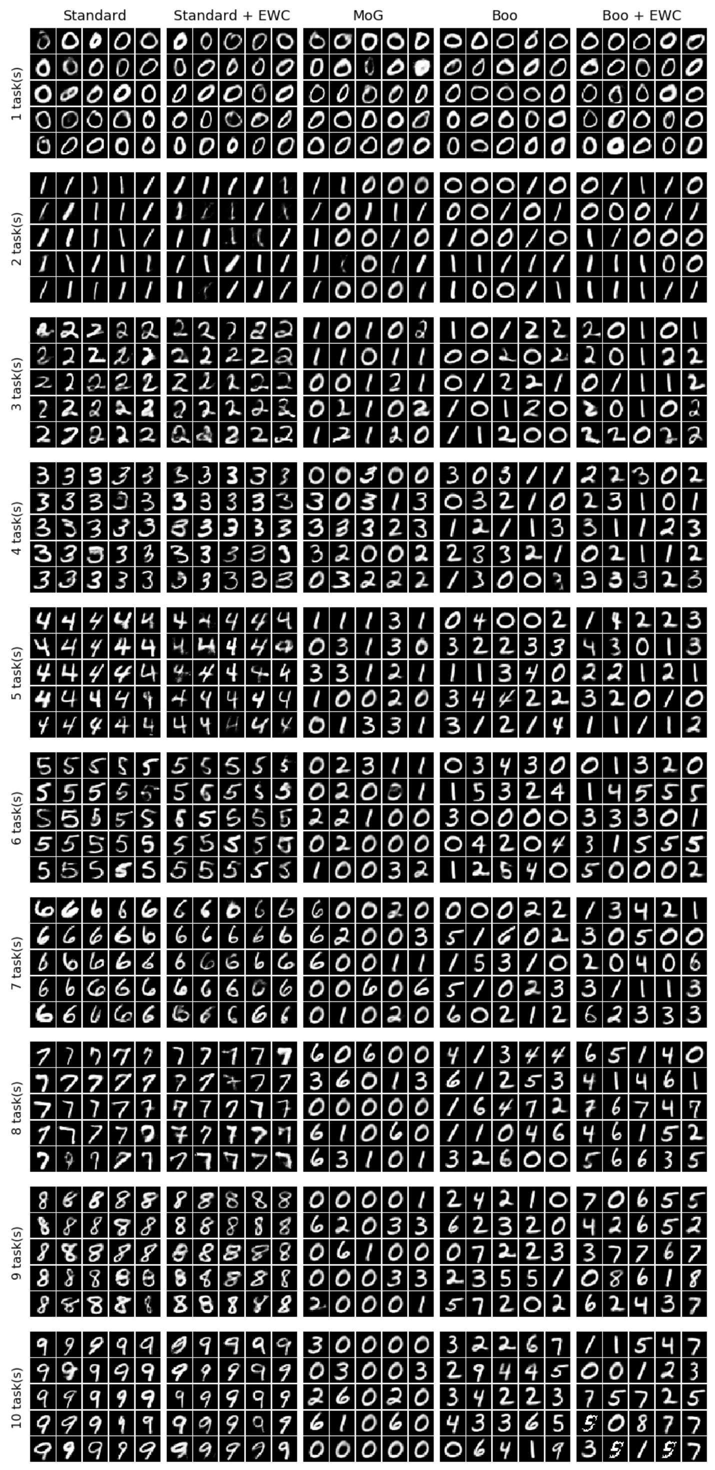


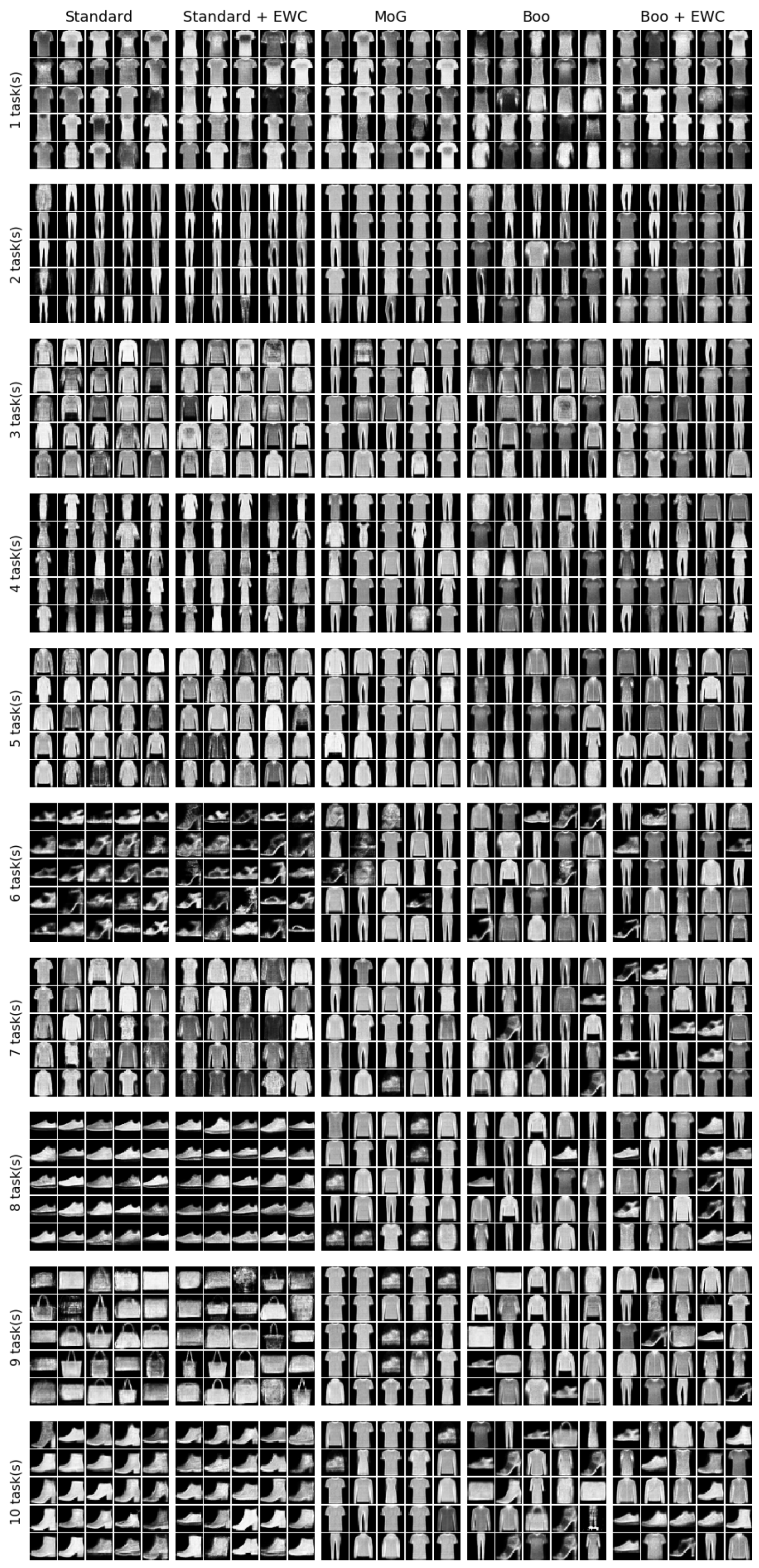
(*b*) fashion MNIST

Figure 3: KL divergence between uniform and generated distribution.

Figure 3 depicts proposed metric to evaluate diversity. We want this value to stay as close as possible to 0 as the number of tasks grows since it will mean that model keeps generating diverse images. We can see a drastic difference between boosting and other approaches: samples from prior, estimated by the boosting approach are very close to uniform in contrast to all the comparable methods.

Appendix F. Examples of generated images




Standard
Standard + EWC
MoG
Boo
Boo + EWC
1 task(s)
2 task(s)
3 task(s)
4 task(s)
5 task(s)
6 task(s)
7 task(s)
8 task(s)
9 task(s)
10 task(s)