Datasets That Are Not: Evolving Novelty Through Sparsity and Iterated Learning

Yusong Wu[1], **Kyle Kastner**[1], **Tim Cooijmans**[1], **Cheng-Zhi Anna Huang**[1,2], **Aaron Courville**[1]
[1]Université de Montréal, Mila [2]Google Brain
wu.yusong@mila.quebec

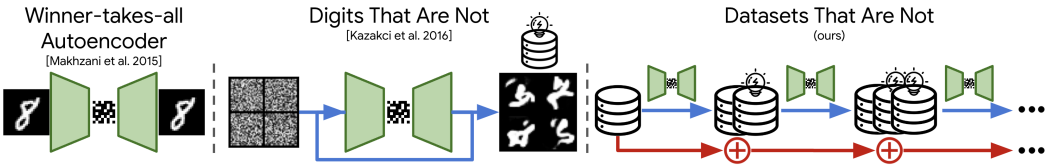

Figure 1: Datasets That Are Not evolves novelty via iterated data accumulation. In each iteration, we first train a convolutional Winner-Take-All autoencoder [1, 2] on the current dataset (leftmost section). After this training stage, we sample novel output from the autoencoder by repeated reconstruction starting from noise (middle section) [2]. We apply iterated data accumulation by adding the generated data from the model to the dataset, which is then used to train a new model at the next stage (rightmost section).

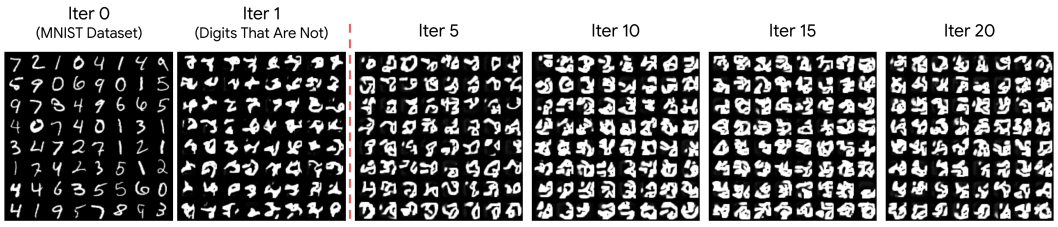

Figure 2: By iteratively accumulating generated data and training new generative models, Datasets That Are Not moves *away* from the styles inferred from generative models trained on only the base MNIST dataset (Iteration 1, far left). We train a new generative model at each iteration in this experiment.

1 Introduction

Creative machines have long been a subject of interest for generative modeling research [3–6]. One research goal of machine creativity is to create machine processes which are data adaptive to develop new creative directions, which may inspire users or be used to provide creative expansions of current ideas. Several works propose models which leverage data-driven deep learning approaches to generate "out-of-domain" or novel samples that deviate from the dataset on which these models are trained [2, 7–10]. In these existing works, generative model weights are only optimized on real datasets, rather than incorporating model generated outputs back into the training loop.

In this work, we propose expanding the scope of a generative model by iteratively training on generated samples, in addition to the given training data. Our approach takes inspiration from the iterative creative flow used by human artists, wherein artists often first learn from others but then iteratively expand or improve on their own past artistic directions, such as previous artwork, prototypes, or sketches. This is similar to techniques from Iterated Learning [11, 12], which uses a broad framework of learning from the output of preceding models, adding previous model outputs into the current training information. Several research works adapt Iterated Learning as an outer loop for machine learning and show improvements in out-of-domain generalization by introducing inductive biases (structure, composition) throughout the iterated learning process [13–15].

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

We study whether a similar process can be effective in a novelty generation setting. In this paper, we propose Datasets That Are Not, a procedure for accumulating generated samples and iteratively training a generative model on this expanding dataset. Specifically, we expand upon Digits that Are Not [2], a sparsity-based autoencoder for the inner generative model, due to the variety and novelty of outputs when trained on the standard MNIST dataset. Our results show that by learning on generated data, the model effectively reinforces its own hallucinations, directing generated outputs in new and unexpected directions *away* from initial training data while retaining core semantics (Section 3).

2 Method

Iterative Data Accumulation We start by training on the MNIST dataset [16]. At each iteration, after the model is trained, we generate a set of new samples using that model as a generator. We then add the generated data into the current dataset, and use the combined real-and-generated dataset to train a new model in the next iteration. This scheme results in an accumulated training dataset which grows larger over training iterations.

Model and Novelty Generation We use Digits That Are Not [2] to generate novel outputs at each stage. The core Digits That Are Not model is a Winner-takes-all (WTA) Autoencoder [1] which consists of a 3-layer convolutional neural network (CNN) [17] encoder and decoder with a sparse bottleneck. In Digits That Are Not, a trained convolutional autoencoder with sparsity bottleneck generates novel samples via repeated reconstruction starting from noise (details in Section A.2). The overall set of generated outputs respect low-level semantics (such as strokes), but deviate in their higher level structure (leftmost figure in Figure 2). During reconstruction only spatial sparsity is activated.

3 Experiments and Results

We use the MNIST dataset as the initial datasource, as in Kazakçı et al. [2]. For sample generation, we set the convergence threshold as 0.001 and the maximum number of steps for sampling each image to 100. For iterated data accumulation, we train the model for 20 iterations, adding a new generated set the same size as MNIST (60000 images) at each iteration.

Model Outputs Deviate over Iterations: Figure 2 shows the result over iterations when we initialize a new model and train from scratch at each iteration. In Figure 2, the model output gradually deviates away from the original output at iteration 1. As iterations increase, the output strokes become thicker and each generated sample has more complex composition of strokes. We believe the thickening of strokes results from sigmoid saturation which is repeatedly strengthened throughout iteration. We also test whether preserving model weights over iterations results in a different bias. Figure 3 shows the result of Datasets That Are Not when keeping the model weights from previous iterations, continually training this model under dataset accumulation. In Figure 3, the model outputs also show a trend of increasing complexity, however this does not increase the stroke thickness as seen in Figure 2.

Data Accumulation is Crucial: We investigate whether data accumulation is crucial in Datasets That Are Not. We run experiments without an accumulating dataset (Figure 4), and "teacher-student" training in order to train on a stream of non-repeated data generated from the previous model (Figure 5). Results in both cases show the degradation and eventual collapse of model output.

Other Generative Models do not Change: We study whether Datasets That Are Not deviates model output when using other generative models which train to generate from the data distribution, instead of focusing on novel output. We test the Datasets That Are Not approach using both generative adversarial networks (GAN) [18] and variational autoencoders (VAE) [19], but find these models only generate digits throughout the iterations (Figure 6 and 7). The novelty and variability of the inner generative model is a crucial aspect of our approach.

Conclusion: We propose a procedure of iteratively adding model generated data into a continually growing dataset for novelty generation. The proposed procedure suggest a way to direct or regularize the model creative output implicitly. Future works include training on different datasets, testing with different novelty generation models, and exploring the effect of filtering the generated data during accumulation.

References

[1] Alireza Makhzani and Brendan J Frey. Winner-take-all autoencoders. *Advances in neural information processing systems*, 28, 2015.

[2] Akın Kazakçı, Cherti Mehdi, and Balázs Kégl. Digits that are not: Generating new types through deep neural nets. In *International Conference on Computational Creativity*, 2016.

[3] Margaret A Boden. Computer models of creativity. *AI Magazine*, 30(3):23–23, 2009.

[4] Simon Colton, Geraint A Wiggins, et al. Computational creativity: The final frontier? In *Ecai*, volume 12, pages 21–26. Montpelier, 2012.

[5] Aaron Hertzmann. Can computers create art? In *Arts*, volume 7, page 18. MDPI, 2018.

[6] Jürgen Schmidhuber. Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. In *Workshop on anticipatory behavior in adaptive learning systems*, pages 48–76. Springer, 2008.

[7] Mehdi Cherti, Balázs Kégl, and Akin Kazakçı. Out-of-class novelty generation: an experimental foundation. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1312–1319. IEEE, 2017.

[8] Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms. *arXiv preprint arXiv:1706.07068*, 2017.

[9] Othman Sbai, Mohamed Elhoseiny, Antoine Bordes, Yann LeCun, and Camille Couprie. Design: Design inspiration from generative networks. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[10] Divyansh Jha, Kai Yi, Ivan Skorokhodov, and Mohamed Elhoseiny. Imaginative walks: Generative random walk deviation loss for improved unseen learning representation. *arXiv preprint arXiv:2104.09757*, 2021.

[11] Kenny Smith, Simon Kirby, and Henry Brighton. Iterated learning: A framework for the emergence of language. *Artificial life*, 9(4):371–386, 2003.

[12] Simon Kirby, Tom Griffiths, and Kenny Smith. Iterated learning and the evolution of language. *Current opinion in neurobiology*, 28:108–114, 2014.

[13] Shangmin Guo, Yi Ren, Serhii Havrylov, Stella Frank, Ivan Titov, and Kenny Smith. The emergence of compositional languages for numeric concepts through iterated learning in neural agents. *arXiv preprint arXiv:1910.05291*, 2019.

[14] Ankit Vani, Max Schwarzer, Yuchen Lu, Eeshan Dhekane, and Aaron Courville. Iterated learning for emergent systematicity in vqa. *arXiv preprint arXiv:2105.01119*, 2021.

[15] Yuchen Lu, Soumye Singhal, Florian Strub, Aaron Courville, and Olivier Pietquin. Countering language drift with seeded iterated learning. In *International Conference on Machine Learning*, pages 6437–6447. PMLR, 2020.

[16] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[19] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[20] Nathan D Ratliff, David Silver, and J Andrew Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27(1):25–53, 2009.

[21] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[22] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3): 56–58, 1997.

[23] Paul Christiano, Buck Shlegeris, and Dario Amodei. Supervising strong learners by amplifying weak experts. *arXiv preprint arXiv:1810.08575*, 2018.

[24] Alisa Liu, Alexander Fang, Gaëtan Hadjeres, Prem Seetharaman, and Bryan Pardo. Incorporating music knowledge in continual dataset augmentation for music generation. *arXiv preprint arXiv:2006.13331*, 2020.

[25] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[28] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in $beta$-vae. *arXiv preprint arXiv:1804.03599*, 2018.

A Appendix

A.1 Related Work

Many methods in machine learning (such as learning-to-search, imitation learning, reinforcement learning, continual learning, and recommender systems) [20–23] share the core idea of accumulating datasets from model output. However, in these areas samples are typically not generated directly using learned models, but instead gathered by interaction with some pre-specified environment or data stream. A closely related generative modeling work is augmentative generation (Aug-Gen) [24], a chorale generation model trained on the works of J.S. Bach. In Aug-Gen, at each iteration the model trains on a dataset of true data and generated samples, generates and filters new samples using a pre-specified grading function, and adds the generated samples which pass the grading filter to the dataset for the next training iteration. Our work differs from Aug-Gen as we target novelty generation by deviating *away* from the training data, and we do not use any grading function or filtering.

A.2 Model and Training Details

The encoder has three 2D convolution layers with kernel size 5 and stride 1. ReLU [25] activation is added between each layer. The decoder is a three-layer 2D transpose convolutional network symmetrical to the encoder, with the same kernel size, stride and ReLU activation function. We set an output channel size of 128 to all the convolutional layers in the autoencoder except the final layer of the decoder, which is set as 1.

The encoder of the autoencoder outputs c channels of 2D feature map. During training, the sparsity bottleneck first keeps the maximum value of the 2D feature map of each channel while masking all other values as 0 (referred to "spatial sparsity" in [1]). Then, for each channel, the batch sparsity keeps top-$k\%$ of the value across batch dimension while masking the 2D feature map of other channels as 0 (referred to "lifetime sparsity" in [1]). We choose a lifetime sparsity of 5% following [1, 2].

We train the autoencoder model using mean square loss (MSE) and the Adam optimizer [26] at a learning rate of 0.001. We use a batch size of 100 to train the autoencoder. We train the autoencoder for 15 epochs on the current dataset.

In Datasets That Are Not training, we use the same model initialization throughout iterations when resetting the model. We also keep the same initial noise input to Digits That Are Not novelty generation for all iterations in one experiment, meaning we use the same initial noise input to generate the samples at each iteration, so any change is solely due to differences in the model weights.

For the teacher-student training setting in Figure 5, we generate a batch of samples from the previous iteration model and directly train the current model using the generated samples. We use the same loss, optimizer setting, and batch size in the teacher-student training. At each iteration, we train the model for 3000 steps.

A.3 Other Experiment Results

Sensitivity to Model Design: We test if small changes in model design change the direction of model output deviation. In this experiment, we tried a variant of the Digits That Are Not model trained with a linear output activation function instead of sigmoid. As the output range is not bounded with linear output activation, we re-normalize the model output after each reconstruction using $X' = \frac{\max(X,0)}{\max(X)}$. As in the previous experiments, we generate and add data of the size of MNIST dataset. Figure 8 shows the result of resetting the model between each iteration, and Figure 9 shows the result of not resetting the model between each iteration.

Compared to Figure 2 using sigmoid activation, the strokes in Figure 8 and 8 are thicker without saturation caused by sigmoid activation. Contrast this with Figure 2, where resetting the model leads to reinforcing the bias of thickening the stroke. The iteration process in Figure 8 is unstable, with fast degradation at first but a non-collapsed result in later iterations.

Keeping the model between each iteration and continuing training is more stable (Figure 9). In that setting, each sample the model outputs shows an increase in the complexity and the number of composed strokes, similar to what has been seen in (Figure 3).

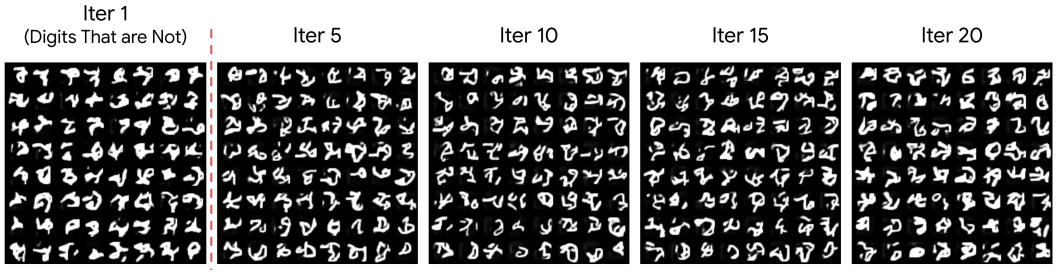

Figure 3: Result of Datasets That Are Not when using an inner Digits That Are Not model, preserving model weights between iterations and adding data of the size of MNIST dataset after each iteration.

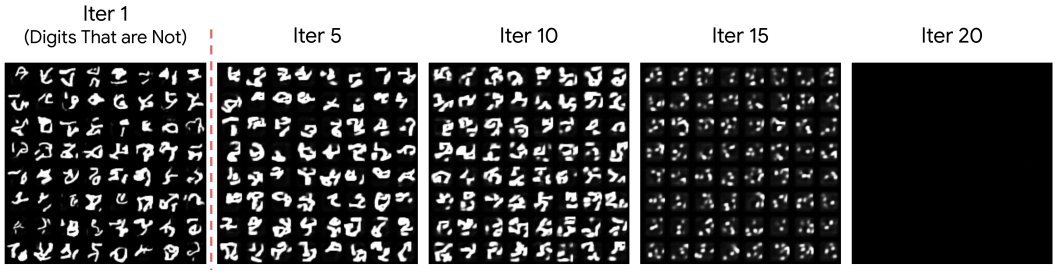

Figure 4: Result of Datasets That Are Not when we do not accumulate data across iteration but generate data the size of the MNIST dataset to pass the model in the next iteration. We reset the model after each iteration in this experiment.

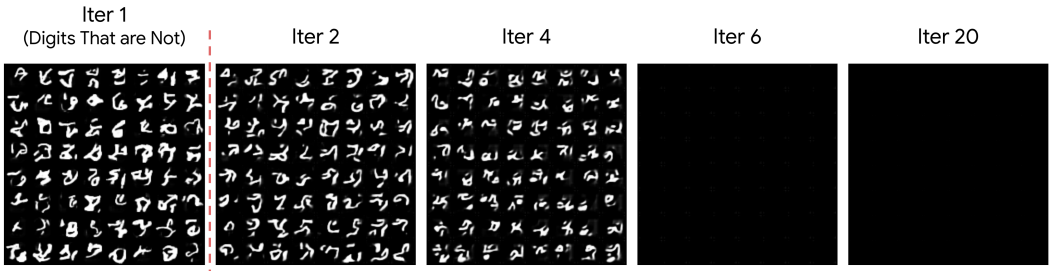

Figure 5: Result of Datasets That Are Not when we do not accumulate data across iteration but generate the data from the preceding model for the model at current iteration to train on in a teacher-student fashion (details in Section A.2). We reset the model after each iteration in this experiment.

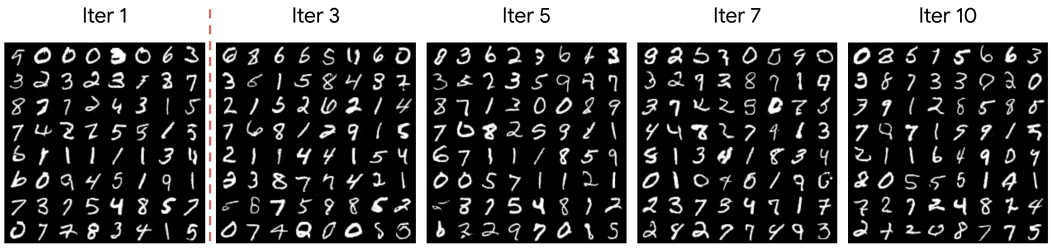

Figure 6: Datasets That Are Not using GAN instead of Digits That Are Not. This experiment follows other settings used in Figure 2 (reset model, add data the size of the MNIST dataset after each iteration). We implement DCGAN [27] for the model in this experiment.

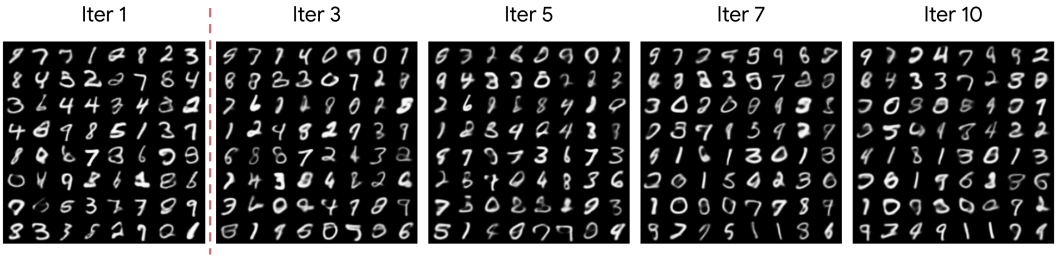

Figure 7: Datasets That Are Not using VAE instead of Digits That Are Not. This experiment follows other settings used in Figure 2 (reset model, add data the size of the MNIST dataset after each iteration). We implement a VAE from Burgess et al. [28] for the model in this experiment.

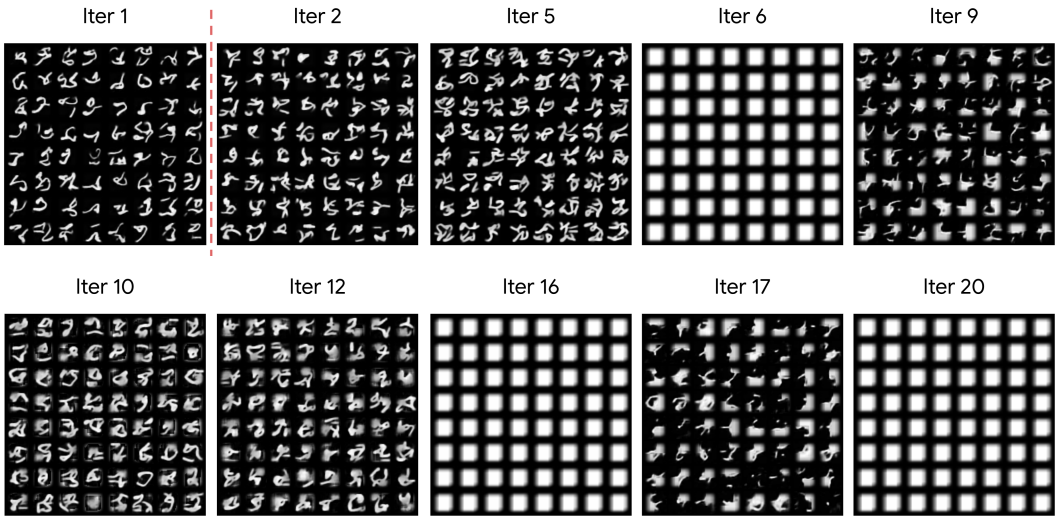

Figure 8: Datasets That Are Not using Digits That Are Not with linear output activation in a Winner-Take-All autoencoder (see Section A.3 for details). In this experiment, we add data the size of the MNIST dataset after each iteration and reset the model between iterations.

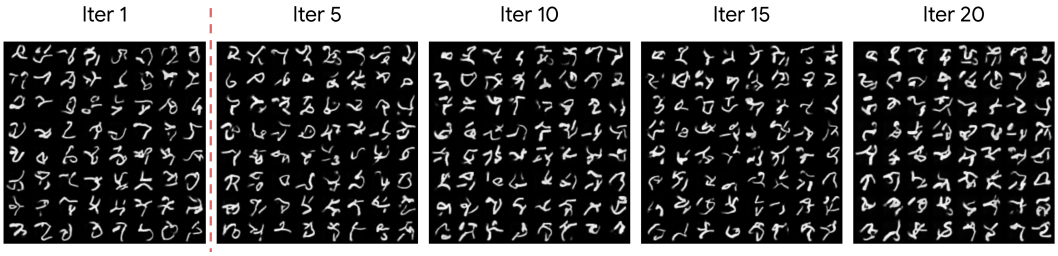

Figure 9: Datasets That Are Not using Digits That Are Not with linear output activation in a Winner-Take-All autoencoder (see Section A.3 for details). In this experiment, we add data the size of the MNIST dataset after each iteration and do not reset the model between iterations.