Out-of-class novelty generation: an experimental foundation *

Mehdi Cherti, Balázs Kégl, Akin Osman Kazakçi

Out-of-class novelty generation: an experimental foundation*

Mehdi Cherti & Balázs Kégl
LAL/LRI
CNRS/Université Paris-Saclay
{mehdi.cherti, balazs.kegl}@gmail.com

Akın Kazakçı
MINES ParisTech,
PSL Research University, CGS-I3 UMR 9217
akin.kazakci@mines-paristech.fr

Abstract

Constructive machine learning aims at finding one or more instances of a domain which will exhibit some desired properties. Such a process bears a strong similarity with a design process where the ultimate objective is the generation of previously unknown and novel objects by using knowledge about known objects. The aim of the present work is to bring ideas from design theory to machine learning and elaborate an experimental procedure allowing the study of design through machine learning approaches. To this end, we propose an actionable definition of creativity as the generation of out-of-distribution novelty. We assess several metrics designed for evaluating the quality of generative models on this new task. Through extensive experiments on various types of generative models, we find architectures and hyperparameter combinations which lead to out-of-distribution novelty. Such generators can then be used to search a semantically richer and broader space than standard generative models would allow.

1 Introduction

Recent advances in machine learning have renewed interest in artificial creativity. Studies such as deep dream [13] and style transfer [4] have aroused both general public interest and have given strong impetus to use deep learning models in computational creativity research [7]. Although creativity has been a topic of interest on and off throughout the years in machine learning [18], it has been slowly becoming a legitimate sub-domain with the appearance of dedicated research groups such as Google's Magenta and research work on the topic [14, 10].

Machine learning methods provide an important advantage for studying computational creativity: they enable the study of creativity in relation with knowledge (i.e., knowledge-driven creativity [9]). Within the scope of machine learning, generative modeling can provide a way to study and answer quetions about novelty generation. According to [6, 8], creativity is about generating previously unknown but meaningful (or valuable) new types of objects using previously acquired knowledge. Under this perspective, the goal of novelty generation is to sample objects from new types. This goal, which we shall call *out-of-distribution generation*, is beyond what can be formalized within the framework of traditional learning theory.

Arguably, the most important problem is the evaluation of what constitutes a good model for generating out-of-distribution. Traditional metrics based on likelihood are of no use since novelty in the out-of-distribution is unlikely by definition. Indeed, research in generative modeling usually aims at eliminating this possibility as this is seen as a source of instability [5, 17] leading to spurious samples [2].

This paper presents a new engineering principle that enables such evaluation, and consequently, rigorous experimental research on the matter: we evaluate the generative potential of models by

*This paper is an adapted version of our submission to ICLR17, available here.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

holding out entire classes, simulating thus unknown but meaningful novelty. The goal of the novelty generator is then to use training classes to build a model that can generate objects from future (hold-out) classes, unknown at training time.

We present three main contributions: First, we design an experimental framework based on hold-out classes to develop and to analyze out-of-distribution generators. Second, we review and analyze the most common evaluation techniques from the point of view of measuring out-of-distribution novelty. We argue that likelihood-based techniques inherently limit exploration and novelty generation. We carefully select a couple of measures and demonstrate their applicability for out-of-distribution novelty detection in experiments. Third, we run a large-scale experimentation to study the ability of novelty generation of a wide set of different autoencoders and generative adversarial networks (GANs).

2 Probabilistic vs. constructive generative models

The generative process is commonly framed in a probabilistic setup: it is assumed that an underlying unknown likelihood model $\mathcal{P}(\cdot)$ should first be learned on an i.i.d. training sample $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, assumed to be generated from $\mathcal{P}(\cdot)$, and then a sampler $\mathcal{S}$ should sample from the learned $\widehat{\mathcal{P}}(\cdot)$. The first step, estimating $\mathcal{P}(\cdot)$ using $\mathcal{D}$, is a classical function learning problem that can be studied through the usual concepts of overfitting and regularization, and algorithms can be designed using the classical train/test principle. The second step, designing $\mathcal{S}$ for sampling from $\widehat{\mathcal{P}}(\cdot)$ is also a classical domain of random sampling with a conceptual framework and a plethora of methods.

Technically both steps are notoriously hard for the high-dimensional distributions and the complex dependencies we encounter in interesting domains. Hence, most of the recent and successful methods get rid of the two-step procedure at the level of algorithmic design, and short-cut the procedure from the probabilistic $\mathcal{D} \to \mathcal{P} \to \mathcal{S}$ to the constructive $\mathcal{D} \to \mathcal{A}$, where $\mathcal{A}(\mathcal{D})$ is a *generator*, tasked to produce sample objects similar to elements of $\mathcal{D}$ but *not identical* to them. $\mathcal{A}$ is fundamentally different from $(\mathcal{P}, \mathcal{S})$ in that there is no explicit fitting of a function, we use $\mathcal{D}$ to directly design an algorithm or a program.

When the probabilistic setup is still kept for analysis, we face a fundamental problem: if we assume that we are given the true likelihood function $\mathcal{P}(\cdot)$, the likelihood of the training sample $\frac{1}{n}\sum_{i=1}^{n} \log \mathcal{P}(\mathbf{x}_i)$ is a random variable drawn independently from the distribution of log-likelihoods of i.i.d. samples of size n, so the trivial generator $\mathcal{A}$ which resamples $\mathcal{D}$ will have the same expected log-likelihood as an optimal i.i.d. sampler. The resampling "bug" is often referred to as "overfitting". While it makes perfect sense to talk about overfitting in the $\mathcal{D} \to \mathcal{P} \to \mathcal{S}$ paradigm (when $\mathcal{P}$ is *fitted* on $\mathcal{D}$), it is somewhat conceptually misleading when there is no fitting step, we propose to call it "memorizing". When a generator $\mathcal{A}$ is trained on $\mathcal{D}$ without going through the fitting step $\mathcal{D} \to \mathcal{P}$, the classical tools for avoiding memorizing (regularization, the train/test framework) may be either conceptually inadequate or they may not lead to an executable engineering design principle.

The conceptual problem of analyzing constructive algorithms in the probabilistic paradigm is not a minor nuisance which can be fixed by augmenting the likelihood to avoid resampling. Rather, it is an inherent property which cannot (or rather, should not) be fixed. The probabilistic framework is designed for generating objects from the distribution of known objects, and this is in an axiomatic contradiction with generating out-of-distribution novelty, objects that are unknown at the moment of assembling a training sample. Resampling (generating exact copies) is only the most glaring demonstration of a deeper problem which is also present in a more subtle way when attempting to generate new *types* of objects.

We are not arguing that the probabilistic generative framework should be banished, it has a very important role in numerous use cases. Our argument is that it is not adequate for modeling out-of-distribution novelty generation. At the algorithmic/computational level the machine learning community has already started to move beyond likelihood. The ingenious idea of using discriminators in GANs [5, 17] is a concrete example; although the setup can be analyzed through the lens of probabilistic sampling, one does not have to fall back onto this framework. If we drop the underlying conceptual *probabilistic* framework, the *constructive* GAN idea may be extended beyond generating from the set which is indistinguishable from the set of existing objects. In the next sections, we will use discriminators to assess the quality of generators whose very goal is to generate novelty: objects that are distinguishable from existing objects.

3 Evaluation of generative models

In our setup, we simulate existing knowledge and novelty by partitioning existing data sets *holding out entire classes*. The goal of the novelty generator is then to use training classes to build a model that can generate objects from future (hold-out) classes, unknown at training. The training classes are digits classes from MNIST and the test classes are letters classes from a dataset we constructed from google fonts (Google fonts). We pre-trained a digit discriminator (on MNIST), a letter discriminator (on Google fonts) and a discriminator on a mixture of digits and letters. We used the discriminators to evaluate novelty generators using different metrics which we outline in the following.

Parzen density estimator Parzen density estimators are regularly used for estimating the log-likelihood of a model [3] when the log-likelihood is not tractable. A kernel density estimator is fit to generated points, and the model is scored by log-likelihood of a hold-out test set under the kernel density. The metrics can be easily fooled [19], nevertheless, we adopted it in this paper for measuring both the in-distribution and out-of-distributions quality of our generators.

Objectness In [17], they proposed an entropy-based score to measure the "objectness" of the generated *set* of objects. This score uses a pre-trained classifier to compute, for each generated object, the confidence of the classifier on each category, then the score is high if the classifier has a highly confident prediction on a category (hence the name objectness), also, the score encourages sets of generated objects which have a diverse set of categories.

Formally, objectness is defined as

$$\frac{1}{N}\sum_{i=1}^{n}\sum_{\ell=1}^{K} p_{i,\ell}\log\frac{p_{i,\ell}}{p_\ell},$$

where K is the number of classes,

$$p_{i,\ell} = \mathcal{P}(\ell|\mathbf{x}_i)$$

is the posterior probability of category ℓ given the generated object $\mathbf{x}_i$, under the discriminator $\mathcal{P}$ trained on a set with known labels, and

$$p_\ell = \frac{1}{n}\sum_{i=1}^{n} p_{i,\ell},$$

are the class marginals.

Out-of-class count and out-of-class max Naturally, letter discriminators see letters everywhere. Since letters are all they know, they classify everything into one of the letter classes, quite confidently (this "blind spot" phenomenon is exploited by [14] for generating "synthetic" novelty), the letter objectness of an in-distribution digit generator can sometimes be high. For example, a lot of *6*s were classified as *b*s. To avoid this "bias", we also trained a discriminator on the union of digits and letters, allowing it to choose digits when it felt that the generated object looked more like a digit. We designed two metrics using this discriminator: *out-of-class count* measures the frequency of confidently classified letters in a generated set, and *out-of-class max* is the mean (over the set) of the probability of the most likely letter. None of these metrics penalize "fixated" generators, outputting the same few letters all the time, so we combine both metrics with a diversity term based on the entropy of the letter posterior (conditioned on being a letter).

Formally, let $p_{i,1},\ldots,p_{i,K_{\text{in}}}$ be the in-class posteriors and $p_{i,K_{\text{in}}+1},\ldots,p_{i,K_{\text{in}}+K_{\text{out}}}$ be the out-of-class posteriors, where $K_{\text{in}} = 10$ is the number of in-class classes (digits), and $K_{\text{out}} = 26$ is the number of out-of-class classes (letters). Let

$$\ell_i^* = \arg\max_{\ell} p_{i,\ell}$$

and

$$\ell_{\text{out}\,i}^* = \arg\max_{K_{\text{in}}<\ell\leq K_{\text{in}}+K_{\text{out}}} p_{i,\ell}$$

be the most likely category overall and most likely out-of-class category, respectively. Let

$$\tilde{p}_\ell = \frac{\sum_{i=1}^{n}\mathbb{I}\{\ell = \ell_{\text{out}\,i}^*\}}{\sum_{i=1}^{n}\mathbb{I}\{\ell_{\text{out}\,i}^* > K_{\text{in}}\}}$$

be the normalized empirical frequency of the out-of-class category ℓ. We measure the diversity of the generated sample by the normalized entropy of the empirical frequencies

$$\text{diversity} = -\frac{1}{\log K_{\text{out}}} \sum_{\ell=K_{\text{in}}}^{K_{\text{in}}+K_{\text{out}}} \tilde{p}_\ell \log \tilde{p}_\ell,$$

and define

$$\text{out-of-class count} = (1-\lambda) \times \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\left\{\ell_i^* > K_{\text{in}} \wedge p_{i,\ell_i^*} > \theta\right\} + \lambda \times \text{diversity},$$

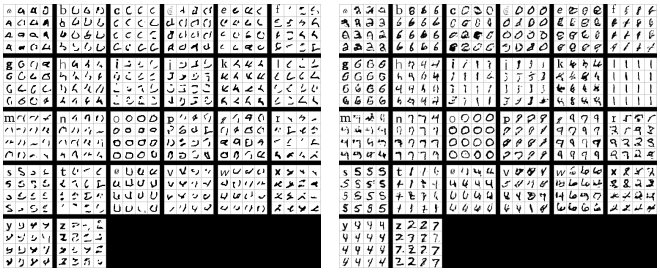
and

$$\text{out-of-class max} = (1-\lambda) \times \frac{1}{n} \sum_{i=1}^{n} p_{i,\ell^*_{\text{out}\,i}} + \lambda \times \text{diversity}.$$

In our experiments we set the confidence level $\theta = 0.95$ and the mixture coefficient $\lambda = 0.5$.

Human evaluation We assessed the visual quality of the set of generated objects using an in-house annotation tool. We took each model which appeared in the top ten by any of the quantitative metrics described in the previous section, and hand-labeled them into one of the following three categories: i) letters, ii) digits, and iii) bad sample (noise or not-a-symbol).

Each panel consisted 26×15 generated objects, the fifteen most probable symbols of each letter according to the classifier trained on both letters and digits (Figure 1). The goal of this annotation exercise was i) to assess the visual quality of the generated symbols and ii) to assess the quality of the metrics in evaluating novelty.



(a) The top autoencoder (b) The top GAN

Figure 1: A couple of the top models according to human assessment. Top left characters of each 4×4 panel are the labels, letters coming from the training sample. For each letter we display the fifteen most probable symbols according to the classifier trained on both letters and digits.

4 Experiments

4.1 Hyperparameter search

We considered two kinds of models, autoencoders and generative adversarial networks (GANs). We used three regularization strategies for autoencoders: sparse autoencoders [11, 12], denoising autoencoders [2] and contractive autoencoders [16]. We generated images from autoencoders using a procedure similar to [1], except that we binarized (deterministically) the images in each iteration. In initial experiments we found that 100 iterations were sufficient for the majority of models to have convergence so we chose to fix the maximum number of iterations to 100. For stochastic gradient optimization of the autoencoder models, we used adadelta [20] with a learning rate of 0.1 and a batch size of 128. We used rectified linear units as an activation function for hidden layers in all models. We use the sigmoid activation function for output layers.

For GANs, we built upon [15] and used their architecture as a basis for hyperparameter search. We modified the code proposed here to sample new combinations of hyperparameters.

Figure 2: A random selection of symbols generated by one of our best autoencoders, the same as the one that generated the letters in Figure 3(b).

We ran a large scale hyperparameter optimization search and evaluated the proposed metrics. All the models were trained on MNIST training data. We obtained nearly 1000 models. We generated 1000 samples from each trained model. We then evaluated all the metrics proposed in section 3 on each trained model by using the 1000 generated samples and the pre-trained discriminators (see section 3). Figure 2 shows samples obtained one of our best autoencoders (a sparse autoencoder trained as in [11]) according to out-of-class count and out-of-class max.

4.2 Analysis

First, we found that tuning (selecting) generative models for in-distribution generation will make them "memorize" the classes they are trained to sample from. This is of course not surprising, but it is important to note because it means that out-of-class generation is non-trivial, and the vast majority of architectures designed and tuned in the literature are not generating out-of-class novelty naturally. Second, we did succeed to find architectures and hyperparameter combinations which lead to out-of-class novelty. Most of the generated objects, of course, were neither digits nor letters (Figure 2), which is why we needed the "supervising" discriminators to find letter-like objects among them. The point is not that *all* new symbols are letters, that would arguably be an impossible task, but to demonstrate that by opening up the range of generated objects, we do not generate noise, rather objects that *can be* forming new categories.

The quantitative goal of this study was to assess the quality of the defined *metrics* in evaluating out-of-distribution generators. We proceeded in the following way. We selected the top ten autoencoders and GANs according to the five metrics of out-of-class (letters) count, out-of-class max, out-of-class objectness, out-of-class Parzen, and in-class Parzen. We then annotated these models into one of the three categories of "letter" (out), "digit" (in), and "bad" (noise or not-a-symbol). The last three columns of Table 1 show that the out-of-class count and out-of-class max scores work well in selecting good out-of-class generators, especially with respect to in-class generators. They are relatively bad in selecting good generators overall. Symmetrically, out-of-class objectness and the Parzen measures select, with high accuracy, good quality models, but they mix out-of-class and in-class generators (digits and letters). Parzen scores are especially bad at picking good out-of-class generators. Somewhat surprisingly, even out-of-class Parzen is picking digits, probably because in-distribution digit generators generate more regular, less noisy images than out-of-class letter generators. In other words, opening the space towards non-digit like "spurious" symbols come at a price of generating less clean symbols which are farther from letters (in a Parzen sense) than clean digits.

We also computed the inter-score correlations in the following way. We first selected the top 10% models for each score because we were after the correlation of the best-performing models . Then we computed the Spearman rank correlation of the scores (so we did not have to deal with different scales and distributions). The first eight columns of Table 1 show that i) in-class and out-of-class measures are anti-correlated, ii) out-of-class count and max are uncorrelated, and are somewhat anti-correlated with out-of-class objectness.

These results suggest that the best strategy is to use out-of-class objectness for selecting good quality models and out-of-class count and max to select models which generate letters. Figure 3 illustrates the results by pangrams (sentences containing all letters) written using the generated symbols. The models (a)-(d) were selected automatically: these were the four models that appeared in the top ten

	inter-score correlations								human counts		
	oc	om	oo	op	ic	im	io	ip	out	in	bad
out count	1	-0.03	-0.13	0.04	-0.12	0.02	-0.07	-0.11	12	0	1 8
out max	-0.03	1	-0.07	0.01	-0.16	-0.10	0.03	-0.09	15	0	5
out objectness	-0.13	-0.07	1	0.21	-0.06	0.08	0.02	-0.08	9	10	1
out Parzen	0.04	0.01	0.21	1	-0.17	0.01	-0.19	-0.20	4	13	3
in count	-0.12	-0.16	-0.06	-0.17	1	0.30	0.1	0.14	-	-	-
in max	0.02	-0.10	0.08	0.01	0.30	1	0.03	0.06	-	-	-
in objectness	-0.07	0.03	0.02	-0.19	0.1	0.03	1	0.00	-	-	-
in Parzen	-0.11	-0.09	-0.08	-0.20	0.14	0.06	0.00	1	0	17	3

Table 1: Inter-score correlations among top 10% models per score and human annotation counts among top twenty models per score. out=letters; in=digits.

both according to out-of-class objectness and out-of-class counts. Letters of the last sentence (e) were hand-picked by us from letters generated by several top models.

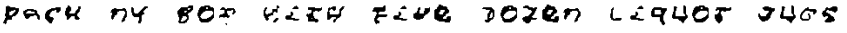
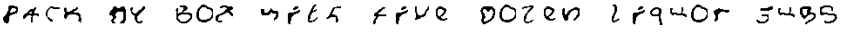
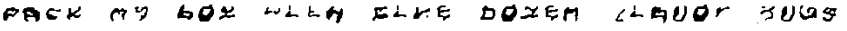
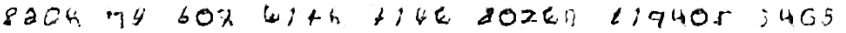
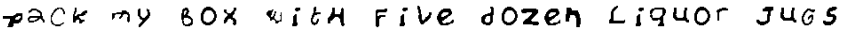
(a) 
(b) 
(c) 
(d) 
(e) 

Figure 3: Pangrams created (a-d) using top models selected automatically, and (e) using letters selected from several models by a human. We note that (b) corresponds to letters selected from the autoencoder in figure 2.

5 Summary and perspectives

The main focus of this paper was setting up the experimental pipeline and to analyze various quality *metrics*, designed to measure out-of-distribution novelty of samples and generative models. The immediate next goal is to analyze the models in a systematic way, to understand what makes them "memorizing" classes and what makes them opening up to generate valuable out-of-distribution samples.

6 Acknowledgments

This work was partially supported by the HPC Center of Champagne-Ardenne ROMEO.

References

[1] D. Bahdanau and H. Jaeger. Smart decisions by small adjust-ments: Iterating denoising autoencoders. Technical report, Technical Report 32, Jacobs University, School of Engineering and Science, 2014.

[2] Y. Bengio, L. Yao, G. Alain, and P. Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pages 899–907, 2013.

[3] O. Breuleux, Y. Bengio, and P. Vincent. Unlearning for better mixing. *Universite de Montreal/DIRO*, 2009.

[4] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[6] A. Hatchuel and B. Weil. Ck design theory: an advanced formulation. *Research in engineering design*, 19(4):181–192, 2009.

[7] ICCC. *Proceedings of the International Conference on Computational Creativity*, 2016.

[8] A. Kazakçı. Conceptive artificial intelligence: Insights from design theory. In *International Design Conference DESIGN2014*, pages 1–16, 2014.

[9] A. Kazakçı, M. Cherti, and B. Kégl. Digits that are not: Generating new types through deep neural nets. In *Proceedings of the Seventh International Conference on Computational Creativity*, 2016.

[10] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[11] A. Makhzani and B. Frey. k-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

[12] A. Makhzani and B. J. Frey. Winner-take-all autoencoders. In *Advances in Neural Information Processing Systems*, pages 2773–2781, 2015.

[13] A. Mordvintsev, C. Olah, and M. Tyka. Inceptionism: Going deeper into neural networks. *Google Research Blog. Retrieved June*, 20, 2015.

[14] A. M. Nguyen, J. Yosinski, and J. Clune. Innovation engines: Automated creativity and improved stochastic optimization via deep learning. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, pages 959–966. ACM, 2015.

[15] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[16] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 833–840, 2011.

[17] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. *arXiv preprint arXiv:1606.03498*, 2016.

[18] J. Schmidhuber. *Driven by Compression Progress: A Simple Principle Explains Essential Aspects of Subjective Beauty, Novelty, Surprise, Interestingness, Attention, Curiosity, Creativity, Art, Science, Music, Jokes*, pages 48–76. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[19] L. Theis, A. v. d. Oord, and M. Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

[20] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.