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

Recent advances in machine learning have brought the field closer to computational creativity research. From a creativity research point of view, this offers the potential to study creativity in relationship with knowledge acquisition. From a machine learning perspective, however, several aspects of creativity need to be better defined to allow the machine learning community to develop and test hypotheses in a systematic way. 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. We also propose a new experimental setup. Inspired by the usual held-out validation, we hold out entire classes for evaluating the generative potential of models. 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 - and thus, are novel with respect to the knowledge the model incorporates. Through extensive experiments on various types of generative models, we are able to find architectures and hyperparameter combinations which lead to out-of-distribution novelty.

1 INTRODUCTION

Recent advances in machine learning have renewed interest in artificial creativity. Studies such as deep dream (Mordvintsev et al., 2015) and style transfer (Gatys et al., 2015) have aroused both general public interest and have given strong impetus to use deep learning models in computational creativity research (ICC, 2016). Although creativity has been a topic of interest on and off throughout the years in machine learning (Schmidhuber, 2009), 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 (Nguyen et al., 2015; Lake et al., 2015).

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 Kazakçı et al. (2016)). Within the scope of machine learning, generative modeling can provide a way to study and answer questions about novelty generation. According to (Hatchuel & Weil, 2009; Kazakçı, 2014), 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 (Goodfellow et al., 2014; Salimans et al., 2016) leading to spurious samples (Bengio et al., 2013).

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 *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 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.

We used Parzen density estimators (Breuleux et al., 2009) and objectness (Salimans et al., 2016). We also propose *out-of-class count* and *out-of-class max* metrics which use a classifier trained on union of digits and letters where the count metric counts the frequency of confidently classified letters in the generated set and the max metric is the mean of the probability of the most likely letter. More details are given in the appendix.

3 EXPERIMENTS

3.1 HYPERPARAMETER SEARCH

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 2 on each trained model by using the 1000 generated samples and the pre-trained discriminators (see section 2). Figure 1 shows samples obtained one of our best autoencoders (a sparse autoencoder trained as in (Makhzani & Frey, 2013)) according to out-of-class count and out-of-class max. More details about the experimental setup is given in the appendix.

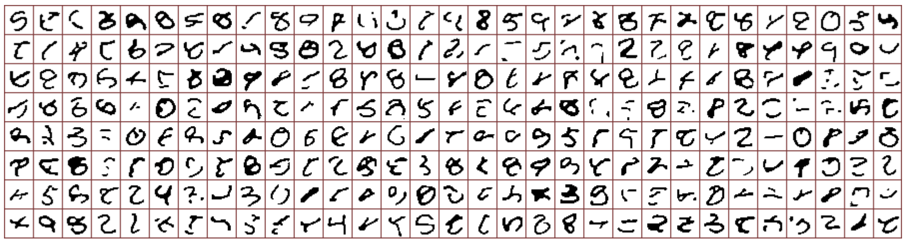

Figure 1: A random selection of symbols generated by one of our best autoencoders, the same as the one that generated the letters in Figure 2(b).

3.2 ANALYSIS

First, we found that tuning (selecting) generative models for in-distribution generation will make them "memorize". 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 1), 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.

	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	18
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.

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 2 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 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.

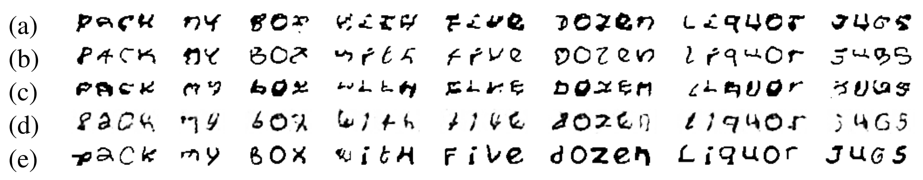

Figure 2: Pangrams created (a-d) using top models selected automatically, and (e) using letters selected from several models by a human. We note that (b) corresponds to letters selected from the autoencoder in figure 1.

4 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.

5 Acknowledgments

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

References

Dzmitry Bahdanau and Herbert Jaeger. Smart decisions by small adjust-ments: Iterating denoising autoencoders. Technical report, Technical Report 32, Jacobs University, School of Engineering and Science, 2014.

Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pp. 899–907, 2013.

Olivier Breuleux, Yoshua Bengio, and Pascal Vincent. Unlearning for better mixing. *Universite de Montreal/DIRO*, 2009.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Armand Hatchuel and Benoit Weil. Ck design theory: an advanced formulation. *Research in engineering design*, 19(4):181–192, 2009.

Proceedings of the International Conference on Computational Creativity, 2016. ICCC.

Akın Kazakçı. Conceptive artificial intelligence: Insights from design theory. In *International Design Conference DESIGN2014*, pp. 1–16, 2014.

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

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Alireza Makhzani and Brendan Frey. k-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

Alireza Makhzani and Brendan J Frey. Winner-take-all autoencoders. In *Advances in Neural Information Processing Systems*, pp. 2773–2781, 2015.

Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. *Google Research Blog. Retrieved June*, 20, 2015.

Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Innovation engines: Automated creativity and improved stochastic optimization via deep learning. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, pp. 959–966. ACM, 2015.

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

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 833–840, 2011.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. *arXiv preprint arXiv:1606.03498*, 2016.

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*, pp. 48–76. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-02565-5. doi: 10.1007/978-3-642-02565-5_4. URL http://dx.doi.org/10.1007/978-3-642-02565-5_4.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

A DETAILS ON THE METRICS

Parzen density estimator Parzen density estimators are regularly used for estimating the log-likelihood of a model (Breuleux et al., 2009) 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 (Theis et al., 2015), nevertheless, we adopted it in this paper for measuring both the in-distribution and out-of-distributions quality of our generators.

Objectness In Salimans et al. (2016), 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 Nguyen et al. (2015) 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

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

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 $\lambda = 0.5$.

B Detailed experimental setup

We used two families of deep learning based generative models, autoencoders and GANs. The architectures and the optional features are described in the next sections. All hyperparameters were selected randomly using reasonable priors. All the $\sim$1000 autoencoders were trained on MNIST training data.

B.1 Autoencoder architectures and generation procedure

We used three regularization strategies for autoencoders: sparse autoencoders (Makhzani & Frey, 2013; 2015), denoising autoencoders (Bengio et al., 2013) and contractive autoencoders (Rifai et al., 2011).

Sparse autoencoders can either be fully connected or convolutional. For fully connected sparse autoencoders, we use the k-sparse formulation from Makhzani & Frey (2013), a simple way of obtaining a sparse representation by sorting hidden units and keeping only the top $k\%$, zeroing out the others, and then backpropagating only through non-zero hidden units.

For convolutional sparse architectures, we use the "winner take all" (WTA) formulation from Makhzani & Frey (2015) which obtains *spatial sparsity* in convolutional feature maps by keeping only the maximum activation of each feature map, zeroing out the others. We optionally combine it with *channel sparsity* which, for each position in the feature maps, keeps only the maximum activation across the channels and zero out the others.

For contractive autoencoders, we use the fully connected version with a single hidden layer from Rifai et al. (2011).

We also explore mixtures between the different autoencoder variants in the hyperparameter search. For each model we choose to enable or disable independently the denoising training procedure, the contractive criterion (parametrized by the contractive coefficient, see (Rifai et al., 2011)) and the sparsity rate k (only for fully connected architectures). Table 2 shows the hyperparameters and their priors.

The generation procedure we use for autoencoders is based on Bengio et al. (2013), who proposed a probabilistic interpretation of denoising autoencoders and a way to sample from them using a Markov chain. To have a convergent procedure and to obtain fixed points, we chose to use a deterministic generation procedure instead of a Markov chain (Bahdanau & Jaeger, 2014). As in Bahdanau & Jaeger (2014), we found that the procedure converged quickly.

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. We also chose to extend the procedure of Bahdanau & Jaeger (2014) by binarizing (using a threshold) the images after each reconstruction step, as we found that it improved the speed of the convergence and could lead to final samples with an exact zero reconstruction error.

For stochastic gradient optimization of the autoencoder models, we used adadelta (Zeiler, 2012) 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.

Table 2: Autoencoder hyperparameter priors.

Name	Prior	Type
nb layers	1, 2, 3, 4, 5	choice
nb fully connected hidden units	100,200,300,...1000	choice
nb conv layers	1, 2, 3, 4, 5	choice
nb conv filters	8, 16, 32, 64, 128, 256, 512	choice
conv layers filter size	3 or 5	choice
noise corruption	[0, 0.5]	uniform
k sparsity rate	[0, 1]	uniform
contraction coefficient	[0, 100]	uniform

B.2 Generative adversarial networks (GANs)

For GANs, we built upon Radford et al. (2015) and used their architecture as a basis for hyperparameter search. We modified the code proposed here to sample new combinations of hyperparameters. Table 3 shows the hyperparameters and their priors.

Name	Prior	Type
nb discr. updates	1, 2, 3	choice
l2 coeficient	$[10^{-6}, 10^{-1}]$	logspace
gen. input dim.	10, 20, 50, 70, 100, 150, 200, 300	choice
nb fully connected gen. units	8, 16, 32, 64, 128, 256, 1024, 2048	choice
nb fully connected discr. units	8, 16, 32, 64, 128, 256, 1024, 2048	choice
nb filters gen.	8, 16, 32, 64, 128, 256, 512	choice
nb filters discr.	8, 16, 32, 64, 128, 256, 512	choice
nb iterations	50, 100, 150, 200, 250, 300	choice
learning rate	$[10^{-6}, 10^{-1}]$ on logspace, or 0.0002	logspace
weight initialization	Normal(0, std) where std is from $[10^{-3}, 10^{-1}]$	logspace

Table 3: GAN hyperparameter priors.