Training Discriminative Models to Evaluate Generative Ones

Timothée Lesort\textsuperscript{1, 2}, Jean-François Goudou\textsuperscript{2}, and David Filliat\textsuperscript{1}

\textsuperscript{1}U2IS, ENSTA ParisTech, INRIA FLOWERS team
\textsuperscript{2}Theresis Laboratory, Thales

June 29, 2018

Abstract

Generative models are known to be difficult to assess. Recent works, especially on generative adversarial networks (GANs), produce good visual samples of varied categories of images. However, the validation of their quality is still difficult to define and there is no existing agreement on the best evaluation process. This paper aims at making a step toward an objective evaluation process for generative models. It presents a new method to assess a trained generative model by evaluating its capacity to teach a classification task to a discriminative model. Our approach evaluates generators on a testing set by using, as a proxy, a neural network classifier trained on generated samples. Neural networks classifier are known to be difficult to train on an unbalanced or biased dataset. We use this weakness as a proxy to evaluate generated data and hence generative models. Our assumption is that to train a successful neural network classifier, the training data should contain meaningful and varied information, that fit and capture the whole distribution of the testing dataset. The success of a proxy classifier will then evaluate the quality of the generator. By comparing results with different generated datasets we are able to classify and compare generative models. The motivation of this approach is also to evaluate if generative models can help discriminative neural networks to learn, i.e., measure if training on generated data is able to make a model successful at testing on real settings. Our experiments compare different generators from the VAE and GAN framework on MNIST and fashion MNIST datasets. The results of our different experiments show that none of the generative models are able to replace completely true data to train a discriminative model. It also shows that the initial GAN and WGAN are the best choices to generate comparable datasets on MNIST and fashion MNIST but suffer from instability. VAE and CVAE are a bit less well-performing but are much more stable.
1 Introduction

Generative models are a particular kind of model that learn to reproduce training data and to generalize it. This kind of model has several advantages, for example as shown in [16], the generalization capacity of generative models have interesting properties and can help a discriminative model to learn by regularizing it. Moreover, once trained, they can be sampled as much as needed to produce new datasets. Generative models such as GAN [6], WGAN [1], CGAN [15], CVAE [25] and VAE [10] have produced samples with good visual quality on various image datasets such as MNIST, bedrooms [19] or imageNet [17]. They can also be used for data augmentation [20], for safety against adversarial example [26], or to produce labeled data [24] in order to improve the training of discriminative models.

One commonly accepted tool to evaluate a generative model trained on images is visual assessment. It aims at validating the realism of samples. However, those methods are very subjective and dependent of the evaluation process. One case of this method is called ‘visual Turing tests’, in which samples are visualized by humans who try to guess if the images are generated or not. It has been used to assess generative models of images from ImageNet [5] and also on digit images [11]. Others propose to replace the human analysis by the analysis of first and second moments of activation of a neural network. This method has been used with the output of the inception model for “Inception Score” (IS) [22], or with activation from intermediate layers for “Frechet Inception Score” (FID) [7]. Log-likelihood based evaluation metrics were also widely used to evaluate generative models but as shown in [13], those evaluations can be misleading in high dimensional cases such as images.

The solution we propose uses the test set of a given dataset to estimate gen-
erated samples quality and global fit to the original data. The testing accuracy of a classifier trained on generated data estimates if a generative model is good at producing a dataset. The full process of the method is illustrated Figure 1.

Our contribution is twofolds: first we propose a method to evaluate generative models on the testing set. Secondly we introduce a quantitative score, the fitting capacity, to evaluate and compare performance of generative models.

2 Related work

2.1 Generative models

Generative models can be implemented in various frameworks and settings. In this paper we explore two kind of those frameworks: variational auto-encoders and generative adversarial networks.

The variational auto-encoder (VAE) framework [10], [21] is a particular kind of auto-encoder which learns to map data into a Gaussian latent space, generally chosen as an univariate normal distribution \( \mathcal{N}(0, I) \) (where \( I \) is the identity matrix). The VAE learns also the inverse mapping from this latent space to the observation space. This characteristic makes the VAE an interesting option for generating new data after training. Indeed, thanks to the inverse mapping, new data can be generated by sampling a Gaussian distribution and decoding these samples. The particularity of the latent space comes from the minimization of the KL divergence between the distribution of the latent space and the prior \( \mathcal{N}(0, I) \). For the sake of simplicity, in this paper we will refer to decoder of the VAE as a generator.

Generative adversarial networks [6] are another framework of models that learn to generate data. The learning process is a game between two networks: a generator that learns to produce images from a distribution \( P \) and a discriminator which learns to discriminate between generated and true images. The generator learns to fool the discriminator and the discriminator learns to not be fooled. This class of generative models can produce visually realistic samples from diverse datasets but they suffer from instabilities in their training. One of the model we evaluate, the Wasserstein GAN (WGAN) [1], try to address those issues by enforcing a Lipschitz constraint on the discriminator. We also evaluate the BEGAN [2], another variant of the GAN which uses an auto-encoder as discriminator.

Both GANs and VAEs can also be implemented into a conditional setting. Conditional neural networks [25] and in particular Conditional Variational Autoencoders (CVAE) and Conditional Generative adversarial networks (CGAN) [15] are a class of generative models that have control over the sample’s class of their training dataset. By imposing a label during training on the generator, a conditional generative network can generate from any class and thus produces labeled data automatically. The conditional approach has been used to improve the quality of generative models and make them more discriminative [18]. They are particularly adapted for our setup because we need to generate
labeled datasets to train our classifiers.

2.2 Evaluation of generated samples

The evaluation of generative models have been addressed in various settings. [13] show that different metrics (as Parzen windows, Nearest Neighbor or Log likelihood) applied to generative models can lead to different results. Good results in one application of a generative model can not be used as evidence of good performance in another application. Their conclusion is that evaluation based on sample visual quality is a bad indicator for the entropy of samples. Conversely, the log-likelihood can be used to produce samples with high entropy and evaluate it but it does not assure good visual quality. In this setting, high entropy means high variability in the samples.

More methods exist to evaluate generative networks as described in [3]. In particular, approaches that use a classifier trained on real data to evaluate generative models, [30, 8]. Different quantitative evaluation have also been experimented in [9] which compares models of GANs in various settings. These quantitative evaluations are based on divergence or distances between real data or real features and generated ones. Our approach is similar to the approach developed in parallel by [23]. However our experiments evaluate generative models with the same generator architecture, all trained by the same method and not models with their original architecture. We compare then the results of different learning criterion with the same models architecture and training process.

2.3 Multi-scale structural similarity

Multi-scale structural similarity (MS-SIM, [27]) is a measurement that gives a way to incorporate image details at different resolutions in order to compare two images. This similarity is generally used in the context of image compression to compare image before and after compression. [18] use this similarity to estimate the variability inside a class. They randomly sample two images of a class and measure the MS-SIM. If the value is high, then images are considered different. By operating this process on multiple data points $X$ of the class $Y$, the similarity gives an insight on the entropy of $P(X|Y)$: if the MS-SIM gives high result, the entropy is high; otherwise, the entropy is low. However, it can not estimate if the sample comes from one or several modes of the distribution $P(X|Y)$. For example, if we want to generate images of cats, the MS-SIM similarity can not differentiate a generator that produces different kinds of black cats from a network that produces different cats of different colors. In our method, if the generator is able to generate in only one mode of the distribution $P(X|Y)$, the score will be low in the testing phase.
2.4 Inception score

One of the most used approach to evaluate a generative model is Inception Score (IS) \cite{22, 18}. The authors use an inception classifier model pretrained on Imagenet to evaluate the sample distribution. They compute the conditional classes distribution at each generated sample \( x \), \( P(Y|X = x) \) and the general classes distribution \( P(Y) \) over the generated dataset.

They proposed the following score:

\[
IS(X) = \exp(\mathbb{E}_X[KL(P(Y|X) \parallel P(Y))] ,
\]

Where \( KL \) is the Kullback-Leibler divergence. The KL term can be rewritten:

\[
KL(P(Y|X) \parallel P(Y)) = H(P(Y|X), P(Y)) - H(P(Y|X)) ,
\]

Where \( H(P(Y|X)) \) is the entropy of \( P(Y|X) \) and \( H(P(Y|X), P(Y)) \) the cross-entropy between \( P(Y|X) \) and \( P(Y) \). The entropy term is low when predictions given by the inception model have high confidence in one class and is high in other cases. The cross-entropy term is low when predictions given by the inception model gives unbalanced classes in the whole dataset and is high if the dataset is balanced.

Hence, the inception score promotes when the inception model predictions have high confidence in varied classes. The hypothesis is that if the inception has high confidence in its prediction the image should look real. Unfortunately, it does not estimate if the samples have intra-class variability (it does not take the entropy of \( P(X|Y) \) into account). Hence, a generator that could generate only one sample per class with high quality would maximize the score.

One important restriction of IS is that the generative models to evaluate should produce images in Imagenet classes.

2.5 Frechet Inception Distance

Another recent approach to evaluate generative adversarial networks is the Frechet Inception Distance (FID) \cite{7}. The FID, as the inception score, is based on low moment analysis. It compares the mean and the covariance of activations between real data (\( \mu \) and \( C \)) and generated data (\( \mu_{gen}, C_{gen} \)). The activation are taken from a inner layer in an inception model. The comparison is done using the Frechet distance (as if the means and covariance where taken from a Gaussian distribution) (see Eq. 3). The inception model is trained on Imagenet.

\[
d^2((\mu, C), (\mu_{gen}, C_{gen})) = ||\mu - \mu_{gen}||^2 + Tr(C + C_{gen} - 2(C \ast C_{gen})^{\frac{1}{2}}) ,
\]

FID have in particular be use in a large scale study to compare generative models \cite{14}.
3 Methods

We evaluate generators in a supervised training setup. The real dataset $D$, the original dataset, is composed of pair of examples $(x, y)$ where $x$ is a data point and $y$ the associated label. The dataset is split in three subset $D_{\text{train}}, D_{\text{valid}}$ and $D_{\text{test}}$ for cross-validation. Our method needs a generative model that can sample conditionally from any given label $y$. This conditional generative model is trained on $D_{\text{train}}$. Then, we sample random labels $\hat{y}$ and use the generative model to get a new dataset $D_{\text{gen}}$ of samples $\hat{x}$. It is used afterwards to train a classifier implemented into a deep neural network. The cross-validation is then applied with $D_{\text{valid}}$ of the dataset. We compare $D_{\text{test}}$ accuracy from training on $D_{\text{gen}}$ with the score of the same classifier model trained only on training data $D_{\text{train}}$ (the baseline). We can summarize our method as follows:

1. Train a conditional generative model over $D_{\text{train}}$
2. Sample data to produce $D_{\text{gen}}$
3. Train a discriminative model (the classifier) over $D_{\text{gen}}$
4. Select a classifier over a validation set $D_{\text{valid}}$.
5. Iterate the process for several generative models and compare the accuracy of the classifiers on $D_{\text{test}}$.

The protocol presented allows to analyze the performance of a model on the whole test set or class by class. As we will see in results, we estimate the stability of the generators by training them with different random seeds.

The simple fact of changing this seeds can have great impact on the generative models training and thus induce a variability in the results. To show that the variability of results come mainly from the instability of the generator and not from the classifier, we compare our results with results computed with a KNN classifiers instead of neural networks. As KNN classifiers are deterministic classifier, if the random seeds produce variability with this kind of classifier, the instability necessarily comes from the generators. The KNN classifier is however not a good option for our evaluation methods because it is not adapted for complex image classification.

The classifier was chosen to be a deep neural network because they are known to be difficult to train if the testing set distribution is biased with respect to the training set distribution. This characteristic is put to good use in order to compare generated dataset and hence generative models. If $D_{\text{gen}}$ contains unrealistic samples or samples that are not variate enough, the classifier will not reach a high accuracy. Moreover to investigate the impact of generated data into the training process of a classifier, we also experiment the method by mixing real data and generated data. The ratio between generated data over the complete dataset is called $\tau$. If $\tau = 0$ there is no generated data and $\tau = 1$ means only generated data.
We call our final score for a generator $G$ the fitting capacity (Noted $\Psi_G$) because it evaluates the ability of a generator to fit the distribution of a testing set distribution. It is the testing accuracy of the classifier $C$ over $D_{Test}$ trained by a generator when $\tau = 1$.

We evaluate models with the generator or discriminator architecture proposed in [4].

4 Experiments

4.1 Implementation details

Generative models are often evaluated on the MNIST dataset. Fashion MNIST [29] is a direct drop-in replacement for the original MNIST dataset but with more complex classes with higher variability. Thus, we use this dataset in order to evaluate different generative models in addition to MNIST.

As presented in the previous section, to apply our method, we need to generate labeled datasets. We used two different methods for that. For the first one, we train one generator for each class $y$ on $D_{train}$. This enables us to generate sample from a specific class and to generate a labeled dataset. In this setting, we compare VAE, WGAN, GAN and BEGAN. The second method uses conditional generative models which can generate the whole labeled dataset $D_{gen}$ directly. In this case, we ran our experiments on CVAE and CGAN. The generators are trained with Adam optimizer on the whole original training dataset for 25 epochs on eight different random seeds.

The classifier model trained on $D_{gen}$ is a standard deep CNN with a softmax output (see in Appendix for details). The classifier is trained with Adam optimizer for a maximum of 200 epochs. We use early stopping to stop the training if the accuracy does not improve anymore on the validation set after 50 epochs. Then, we apply model selection based on the validation error and compute the test error on $D_{test}$. The architecture of the classifier is kept fixed for all experiments.

The experiment made with various values of $\tau$ evaluate the results for $\tau = [0.000, 0.125, 0.250, 0.375, 0.500, 0.625, 0.750, 0.875, 1.000]$. $\tau = 0$ is used as a baseline to compare other results. However most of the analysis are made on $\tau = 1$ because the results are representative of the full quality of the generator, i.e. generalization and fitting of the distribution of the testing set.

The experiment with various $\tau$ makes it possible to visualize if a generator is able to generalize and fit a given dataset. If the method results improve when $\tau > 0$ it means that the generator is able to realize data augmentation i.e. generalize training data and if the result are as good when $\tau$ is near 1 than when $\tau$ is near 0 it means the generator is able to fit $D_{test}$ distribution as well as the original training set.
4.2 Adaptation of Inception Score and Frechet Inception Distance

We compare our evaluation method of the generative model with the two most used methods: IS and FID. IS and FID, as originally defined, use a model pretrained on Imagenet. To apply those method, it is mandatory to use the exact same model proposed in [22] with the exact same parameter because in other case results are not comparable with results from other paper. However, as proposed in [12] we can adapt those methods to other setting (for us, MNIST and Fashion-MNIST) to compare several generative models with each other.

We therefore train a model for classification on $D_{train}$. Then we use it to produce a probability vector to compute IS and an activation vector to compute FID. The activation are from a layer in the middle of the model (details in appendix). The very same formula than IS and FID can then be used to compare models. As described previously, the results produced here are not comparable with results presented in other papers since they are from another model but they can be used to compare the generative models presented in this paper with each other.

4.3 Results

The relevant results presented below are the maximum fitting capacity of each model over all seeds in order to evaluate what models can achieve and statistics on the results among those 8 seeds to give insight on the stability of each model with regards to the random seed.

First we present boxplots of fitting capacity results of each models in Figures 2a and 2b. They present the median value along with the first and last quartile limits. Furthermore they display the outliers of each training (values that are outside 1.5 of the interquartile range (IQR) over the different seeds). This representation is less sensible to outliers than mean value and standard deviation.
Table 1: Mean $\Psi_G$

| Datasets | Baseline | VAE  | CVAE | GAN  | CGAN | WGAN | BEGAN |
|----------|----------|------|------|------|------|------|------|
| MNIST    | 97.81%   | 96.39% | 95.86% | 86.03% | 96.45% | 94.25% | 95.49% |
| Fashion  | 86.59%   | 80.75% | 73.43% | 67.75% | 77.68% | 73.43% | 77.64% |

Table 2: Best $\Psi_G$

| Datasets | Baseline | VAE  | CVAE | GAN  | CGAN | WGAN | BEGAN |
|----------|----------|------|------|------|------|------|------|
| MNIST    | 97.94%   | 96.82% | 96.21% | 97.18% | 96.45% | 97.37% | 95.86% |
| Fashion  | 87.08%   | 81.85% | 81.93% | 84.43% | 78.63% | 81.32% | 81.32% |

Those results show an advantage for the GAN model as it has the best median value (even if it does not make better than baseline). Unfortunately some of the generator training failed (in particular on Fashion-Mnist), producing outliers in the results. WGAN produce results comparable to GAN in MNIST but seems less stable in our setting.

The figures are complemented with the values computed for the mean fitting capacity $\Psi_G$ in Table 1 and the values for the best fitting capacity in Table 2. We can note that for both MNIST and Fashion-MNIST, models with unstable results, as GAN and WGAN, have the best $\Psi_G$ however since some training failed for GAN and WGAN, more stable models, as VAE and CGAN, have the best mean fitting capacity.

In addition, we present in Figures 3a and 3b, the per class fitting capacity. The figures show the difference between the baseline classes results and the classifier trained on the generator to evaluate. For generative models that are not conditional and trained class by class, those figures show how the generator is successful to generate in different classes of the dataset. For conditional generative models, it evaluates if the models is able to learn each mode of the distribution with the same accuracy.

We can also estimate the stability of each models class by class or from a class to another. For example we can see that WGAN on MNIST is very stable except on class 1 and on Fashion-MNIST it seems to struggle a lot between the first 3 classes. On an other hand we can see that BEGAN has some trouble on Fashion-MNIST on class 0 and 2 (T-shirt and pullover) suggesting that the generator is not good enough to discriminate between those two classes.

In Figure 4, we present the test accuracy with various $\tau$ to represent the impact of generated data on the training process. When $\tau = 0$, there is no generated data, this is the result of the baseline. Figures 4b and 4a show result of cherry-picking the best results for each $\tau$ and seed, while Figures 4c and 4d show statistics among different seeds and the stability of each model on the trained dataset. Our interpretation is that if the accuracy is better than baseline

---

1The Figures 2a and 2b are zoomed to be able to visually discriminates models making some outliers out of the plot. Full figures are in appendix.
Figure 3: Plot of the difference between models performance and baseline class by class when \( \tau = 1 \): Mean and standard deviation over random seeds
with a low $\tau$ ($0 < \tau < 0.5$) it means that the generator is able to generalize by learning meaningful information about the dataset. When $\tau > 0.5$ if the accuracy is still near the baseline it means the generated data can replace the dataset in most parts of the distribution. When $\tau = 1$, the classifier is trained only on generated samples. If the accuracy is still better than the baseline, it means that the generator has fitted the training distribution (and eventually has learned to generalize if this score is high over the test set).

Our results show that some models are able to produce data augmentation and outperform the baseline when $\tau$ is low but unfortunately none of them is able to do the same when $\tau$ is high. This shows that they are not able to replace completely the true data in this setting. Following this interpretation, Figure 4 allows us to compare different generative neural networks on both datasets. For example, we can see that all models expectation are equivalent when $\tau$ is low (Figure 4d and 4c). However when $\tau$ is high we can clearly differentiate generative models type.

Some of the curves in Figure 4c and 4d have high standard deviation (e.g.
for GAN when $\tau = 1$). To show that it is not due to the classifier instability we plot the results of classification with various $\tau$ with a KNN classifier (Figure 5a and 5b (k=1)). KNN algorithms are stable since they are deterministic. The standard deviations found with KNN classifiers is similar to those with neural networks classifiers. This proves that the instability does not come from the classifier but from the generative models. This is coherent with the fact that in Figure 2a and 2b the diagrams of the reference classifiers trained with true data on eight different seeds show a high stability of the classifier model.

4.4 Comparison with IS and FID

We compare our results to IS and FID methods. The two methods have been slightly adapted to fit our setting as described previously in Section 3. To be able to compare easily the different methods, we normalize values in order to have a mean of 0 and a standard deviation of 1 among the different models. Originally for FID methods, a low value means a better model, for easier comparison we multiply the FID score by $-1$ to valuate by an higher value a better model as for the other evaluation methods. The results are shown on Figure 8a and 8b. We added a baseline for each method, for the fitting capacity the baseline is the test accuracy when the classifier trained on true data, the inception score baseline is computed on the test data and the frechet inception distance baseline is computed between train data and test data.

The results of the inception score are completely different between MNIST and Fashion-MNIST and some model radically beat the baseline (as WGAN) when in the other methods, none of the results outperform the baselines. However, the Frechet inception score computed between test set and generated data gives coherent results between MNIST and fashion-MNIST. Even if they are not always coherent with our results. As an example, VAE does not perform well with FID when we can see with our fitting capacity that it is able to train a classifier quite well with high stability. The FID baseline outperform the other
models however there is a small margin between best model and baseline when the margin between best model and baseline is big for the fitting capacity. The performances of each model are unfortunately specific to each datasets and the experiments made are not sufficient to generalized results to other datasets. This can be seen in models like CGAN or WGAN where results are very different between MNIST and Fashion-MNIST.

Those results show a high instability for certain models and therefore should be interpreted carefully. Different random seeds would gives different results. However, they show that it is possible to evaluate and compare generative models on their top performance and there stability with regards to random seeds as well as detect failing training. We hope that the use of our method will be a step for clear evaluation and comparison between generative models.

5 Discussion

We presented a method to evaluate a generative model on the testing set of a labeled dataset. It assesses how well the generative model learned to generalize and fit a distribution in a conditional setting. The use of a discriminative model to evaluate generative models and GANs in particular, have already be experimented to compare generative models (IS, FID). However, the model used in those methods is pretrained on labeled data and dependent on the classifier proposed. This is problematic because the choice of the model is biased with respect to this classifier. As a testing set is specifically designed to measure the ability to understand data it is therefore more representative than the analysis of first and second moments of activation in an empirically chosen model to evaluate a generated dataset quality. Furthermore our approach relies only on the testing set a not on generative model or classifier architecture for results comparison and classification.
In this paper, we applied this method on images samples. It means that we could compare our measure with a visual assessment. Our assessment method can also be used with other kind of data as long as labeled data are available to train and evaluate generators.

Unfortunately our evaluation method takes time with regards to other methods because a classifier needs to be trained. However, we can take advantage from both our method and other at the same time for example by applying model selection with FID during training and applying our method for deeper analysis.

Our evaluation was performed on several seeds, datasets and generative models. With the current results, GAN and WGAN seems to be the most efficient solutions. However, this result should be confirmed by experiments on other datasets, generative models and with different types of discriminative models to get a more general comparison.

As presented in [28], the sampling method of a generator can be optimized, which can have an impact on the data produced. A way to boost the performance of a generator can be to focus on improving the sampling phase instead of the model design or the training. An extension of this work could be to look into the impact of several sampling techniques on the performance of a generative model.

6 Conclusion

This paper introduces a method to assess and compare the performances of generative models. By training a classifier on generated sample we can estimate the ability of a generative model to fit and generalize a testing set. It does not directly assess the realistic characteristics of the generated data but rather if their content and variability contains enough information to classify real data.

This method makes it possible to take into account complex characteristics of the generated samples and not only the distribution of their features. Moreover it does not evaluate generative models by testing if we can discriminate true data from generated one. Our results suggest that to get the best results GAN or WGAN approach should be privileged, however to maximize the chance of having a decent result CGAN or VAE are preferable.

In order to have a fair comparison we use same classifier to evaluate all generative models. However, any classifier could be used in replacement to improve and complete the results, our focus is on keeping the same test set to compare models. For instance it is obvious that a generator that just learned to reproduce the training set will beat our results, since no generator are able to beat the baseline. However we hope that new generators or better classifier will be able to beat the baseline. This work is not a comprehensive evaluation of generative models but we believe that it is simple enough to be easily reproducible to evaluate any generative model.

An example of application of generative model used as replacement of true data is embedded platform. An autonomous agent can save a generative model
to reproduce training data in order to save memory space, for example fashion-MNIST length is 55MB when our model of CVAE is only 26.7MB. As a future work, our approach could be used to optimize the generator size while keeping the ability to reproduce sufficient data to train a classifier.

Acknowledgement

We really want to thanks Florian Bordes for experiment settings and interesting discussions as well as Pascal Vincent for his helpful advises. We would like also to thanks Natalia Díaz Rodriguez and Anthonin Raffin for their help in proof reading this article.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In Proceedings of the 34th International Conference on Machine Learning, 2017.

[2] D. Berthelot, T. Schumm, and L. Metz. BEGAN: boundary equilibrium generative adversarial networks. CoRR, abs/1703.10717, 2017.

[3] A. Borji. Pros and Cons of GAN Evaluation Measures. ArXiv e-prints, Feb. 2018.

[4] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. CoRR, abs/1606.03657, 2016.

[5] E. L. Denton, S. Chintala, a. szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In Advances in Neural Information Processing Systems 28. 2015.

[6] 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 27. 2014.

[7] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. ArXiv e-prints, June 2017.

[8] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. CoRR, abs/1611.07004, 2016.

[9] D. Jiwoong Im, H. Ma, G. Taylor, and K. Branson. Quantitatively Evaluating GANs With Divergences Proposed for Training. ArXiv e-prints, Mar. 2018.
[10] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In Proceedings of the 2nd International Conference on Learning Representations (ICLR), 2014.

[11] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. Science, 2015.

[12] C. Li, H. Liu, C. Chen, Y. Pu, L. Chen, R. Henao, and L. Carin. Alice: Towards understanding adversarial learning for joint distribution matching. Neural Information Processing Systems (NIPS), 2017.

[13] A. v. d. O. Lucas Theis and M. Bethge. A note on the evaluation of generative models. In Proceedings of the 4th International Conference on Learning Representations (ICLR), 2016.

[14] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are GANs Created Equal? A Large-Scale Study. ArXiv e-prints, Nov. 2017.

[15] M. Mirza and S. Osindero. Conditional generative adversarial nets. CoRR, abs/1411.1784, 2014.

[16] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Advances in neural information processing systems, 2002.

[17] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[18] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier GANs. In Proceedings of the 34th International Conference on Machine Learning, 2017.

[19] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. CoRR, abs/1511.06434, 2015.

[20] A. J. Ratner, H. R. Ehrenberg, Z. Hussain, J. Dunnmon, and C. Ré. Learning to compose domain-specific transformations for data augmentation. stat, 2017.

[21] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In Proceedings of the 31th International Conference on Machine Learning,, 2014.

[22] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. CoRR, abs/1606.03498, 2016.
[23] S. Santurkar, L. Schmidt, and A. Madry. A classification-based perspective on GAN distributions, 2018.

[24] L. Sixt, B. Wild, and T. Landgraf. Rendergan: Generating realistic labeled data. CoRR, abs/1611.01331, 2016.

[25] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In Advances in Neural Information Processing Systems 28, 2015.

[26] W. Wang, A. Wang, A. Tamar, X. Chen, and P. Abbeel. Safer Classification by Synthesis. ArXiv e-prints, Nov. 2017.

[27] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In Proc 37th Asilomar Conf on Signals, Systems and Computers, 2003.

[28] T. White. Sampling Generative Networks. ArXiv e-prints, Sept. 2016.

[29] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[30] R. Zhang, P. Isola, and A. A. Efros. Colorful Image Colorization. ArXiv e-prints, Mar. 2016.
A Additional results

B models

B.1 Generator Architectures

Table 3: VAE and GAN Generator Architecture

| Layer | Architecture |
|-------|--------------|
| 1     | FC (20,1024) + BN + relu |
| 2     | FC (1024,128*7*7) + BN + relu |
| 3     | ConvTranspose2d(128, 64, 4, 2, 1) + BN + relu |
| 4     | nn.ConvTranspose2d(64, 20, 4, 2, 1) + sigmoid |

B.2 Classifier Architectures

Table 4: MNIST

| Layer | Architecture |
|-------|--------------|
| 1     | conv(5*5), 10 filters + maxpool(2*2) + relu |
| 2     | conv(5*5), 20 filters + maxpool(2*2) + relu |
| 3     | dropout(0.5) |
| 4     | FC (320, 50) + relu |
| 5     | FC (50, 10) + log-softmax |

Table 5: Fashion-MNIST

| Layer | Architecture |
|-------|--------------|
| 1     | conv(5*5), 16 filters + maxpool(2*2) + relu |
| 2     | conv(5*5), 32 filters + maxpool(2*2) + relu |
| 3     | dropout(0.5) |
| 4     | FC (512, 10) + log-softmax |
Figure 7: Analysis and comparison of models when $\tau = 1$ with all outliers

Figure 8: Comparison between results with mean and standard deviation over random seeds from different approaches. Each result have been normalize to have mean=0 and standard deviation=1 among all models.

**FID Note:** The activation vector to compute FID have been taken arbitrary at the output of layer 2 for both MNIST and Fashion-MNIST