Sampling Prediction-Matching Examples in Neural Networks: A Probabilistic Programming Approach

Serena Booth*, **Ankit Shah***, **Yilun Zhou***, **Julie Shah**
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{serenabooth, ajshah, yilun, julie_a_shah}@csail.mit.edu

Abstract

Though neural network models demonstrate impressive performance, we do not understand exactly how these black-box models make individual predictions. This drawback has led to substantial research devoted to understand these models in areas such as robustness, interpretability, and generalization ability. In this paper, we consider the problem of exploring the prediction *level sets* of a classifier using probabilistic programming. We define a prediction level set to be the set of examples for which the predictor has the same specified prediction confidence with respect to some arbitrary data distribution. Notably, our sampling-based method does not require the classifier to be differentiable, making it compatible with arbitrary classifiers. As a specific instantiation, if we take the classifier to be a neural network and the data distribution to be that of the training data, we can obtain examples that will result in specified predictions by the neural network. We demonstrate this technique with experiments on a synthetic dataset and MNIST. Such level sets in classification may facilitate human understanding of classification behaviors.[1]

Introduction

Are we comfortable with neural networks—with their high-dimensional, uninterpretable intermediary representations—making high stakes decisions such as who is released from jail or who receives a mortgage? As neural networks gain traction in commercial products, enabling people to understand these models becomes essential. To this end, many research communities have analyzed neural network properties. Research on fairness of neural networks aims to ensure models do not discriminate against protected groups (Corbett-Davies et al., 2017). Research on robustness analyzes whether a model will perform drastically differently if the input is only slightly different from test data (Carlini et al., 2019). Research on interpretability tries to understand how decisions are made (Doshi-Velez and Kim, 2017). Finally, research on transfer learning explores if a neural network trained on one data distribution can work well or be adapted to work well on another distribution at test time (Torrey and Shavlik, 2010).

To complement these research approaches, we propose a technique for analyzing sets of examples of a given prediction confidence to evaluate a network's global performance. Our technique empirically evaluates prediction *level sets*, or sets of a given confidence, through applying probabilistic programming (Cusumano-Towner et al., 2019). We first create a generative model and train a neural network to classify generated images. We then sample examples of a given confidence by applying Metropolis Hastings to infer a latent representation of an image which meets the specified classification confidence. We demonstrate our technique on two domains: a procedurally-generated synthetic domain and MNIST digits produced with a VAE or a GAN. Our results show that our inference successfully samples a set of examples for the given classification confidence in each domain. We further demonstrate how our technique can sample "ambiguous" images from network decision boundaries.

Related Work

Adversarial Attacks on Neural Networks

Many approaches demonstrate that neural network predictions are locally unstable; i.e. an input can be slightly perturbed to make the network produce vastly different predictions. Szegedy et al. (2013) first demonstrated that images of seemingly random noise can receive confident predictions and a correctly classified image can be slightly perturbed to be confidently predicted as an incorrect class. Subsequent work has focused on more effective attack methods (e.g. Goodfellow, Shlens, and Szegedy, 2014; Nguyen, Yosinski, and Clune, 2015; Carlini and Wagner, 2017; Athalye et al., 2017; Li et al., 2018; Athalye, Carlini, and Wagner, 2018) and better defenses against them (e.g. Madry et al., 2017; Cohen, Rosenfeld, and Kolter, 2019; Lecuyer et al., 2019).

Remarkably, all these adversarial attacks produce examples that are out of distribution, and it is hard to characterize the distributional properties of such adversarial examples. Our work instead aims to produce "adversarial" examples in some *known* distribution implicitly specified through a generative process. Our method is able to generate examples with arbitrary predictions (e.g. equally confident predictions across classes), while most adversarial attacks generate ex-

*Equal contribution

[1]Code: https://github.com/serenabooth/level-set-inference

amples with confident predictions but incorrect labels. If the data generation models the distribution of the whole dataset except for a class l, and we generate examples according to that distribution but also enforce confident prediction in class l, we can uncover in-distribution adversarial examples.

Density Estimation

Estimating the data distribution is an important and challenging task. Traditionally, parametric (e.g. Gaussian mixture model) and non-parametric (e.g. kernel density estimation (Rosenblatt, 1956)) methods work well on low dimensional data, in which different features represent distinctively different meanings (such as age, salary, height, etc.). However, recent deep learning applications work on very high-dimensional datasets in which features are also highly correlated with each other. For example, an image can easily have tens of thousands of features (i.e. pixels), and nearby pixels are correlated. It is thus much more difficult, if even possible, for these traditional models to capture the data distribution and generate realistic-looking data.

A commonly used model for estimating such high dimensional data distributions is an autoencoder, in which two neural networks respectively encode (compress) and decode (reconstruct) the image into and from a low-dimensional latent vector. Variational autoencoders (VAEs) (Kingma and Welling, 2013) additionally regularize the latent vectors to conform to a unit Gaussian distribution, which reduces the task of sampling from the data distribution to reconstruction from unit Gaussian samples. However, due to the training objective of minimizing l_2 loss, the images reconstructed for vision domain problems through a VAE are typically less sharp than the original images.

Compared to autoencoders, generative adversarial networks (GANs) (Goodfellow et al., 2014) employ a generator-discriminator architecture designed to specifically regularize the generated image to stay within the data distribution. Consequently, mode collapse, in which the generated data lacks the variety found in the original data, is a common problem, and several methods (e.g. Srivastava et al., 2017; Lin et al., 2018) have been proposed to solve the problem. In our experiments on a synthetic dataset with known low-dimensional features, we use the groundtruth distribution. On the MNIST image dataset, we use both a variational autoencoder and a GAN to generate images from the respective learned data distributions.

Confidence in Neural Networks

Our technique explores confidence level sets in neural networks. However, prior work has shown many modern neural network architectures result in overconfident networks, with many incorrect predictions having undue high confidence (Guo et al., 2017). To address the problem of overconfidence in neural networks, Lee et al. (2017) used a GAN to generate out-of-distribution data and regularize the classifier to have uniform confidence (i.e. complete ambivalence) on these examples. Several Bayesian formulations have also been formulated with different approximate inference methods (e.g. Blundell et al., 2015; Graves, 2011; Balan et al., 2015). Gal and Ghahramani (2016) drew a connection between the dropout layer (Srivastava et al., 2014) and Bayesian formulation and used this to calculate uncertainty. Lakshminarayanan, Pritzel, and Blundell (2017) proposed an ensemble approach to capture model uncertainty.

Our work is complementary to these models in that we do not attempt to alter the confidence of a trained neural network model, but instead expose examples with particular confidence values. As a result, if the model is over-confident, our sampling approach may return few or even no valid examples, within some specified distribution. Moreover, since we do not require access to anything beyond a prediction confidence (e.g. gradient information), our model can be used to study all of these confidence-calibrated models.

Probabilistic Programming

Probabilistic programming consists of the emerging set of tools in which programming languages enable the definition of models and automate parts of inference. With popularity of deep learning, several frameworks (e.g. Bingham et al., 2018; Salvatier, Wiecki, and Fonnesbeck, 2016; Tran et al., 2016) also have deep learning integration, in which a neural network model is used to define and/or learn a distribution. In our work, we use the state-of-the-art language Gen (Cusumano-Towner et al., 2019). One prior application of probabilistic programming has parallels to our approach: Fremont et al. (2018) designed a constrained probabilistic programming language, SCENIC, which generates distributions over scenes produced from a generative model. While our work is inspired by SCENIC, their system was designed to explore clear classification *failures* and assist in dataset augmentation, whereas we apply probabilistic programming to explore confidence level sets—including classification successes.

Methodology

Given a classifier $f : X \to \Delta^L$ that maps a data point to the probability simplex of L classes, the overall goal is to find an input $\boldsymbol{x} \in X$ such that $f(\boldsymbol{x}) = \boldsymbol{p}$ for some particular prediction confidence $\boldsymbol{p} \in \Delta^L$. A common approach to achieve this is to start with some initial guess $\boldsymbol{x}_0$, and iteratively optimize the function $\mathcal{L}(x) = d(f(\boldsymbol{x}), \boldsymbol{p})$ using gradient-based method, for some distance metric $d(\cdot, \cdot)$. One such example distance metric is the total variation distance. This method is reminiscent of popular adversarial attacks. However, there is no guarantee that such the resulting $\boldsymbol{x}^*$ will still remain in the original data distribution.

Thus, we introduce a data distribution $p(\boldsymbol{x})$ and consider the inference problem of sampling from the posterior

$$p(\boldsymbol{x}|f(\boldsymbol{x}) = \boldsymbol{p}) \propto p(\boldsymbol{x})p(f(\boldsymbol{x}) = \boldsymbol{p}|\boldsymbol{x}).$$

Note that the likelihood is actually a delta function; this makes sampling infeasible because in general the set $\{\boldsymbol{x} : f(\boldsymbol{x}) = \boldsymbol{p}\}$ has measure 0. To solve this problem, we relax the formulation by introducing $\tilde{\boldsymbol{p}}_{\boldsymbol{x}}$, which is sampled from a Dirichlet distribution with parameters proportional to $f(\boldsymbol{x})$:

$$\tilde{\boldsymbol{p}}_{\boldsymbol{x}} \sim \mathrm{Dir}(\alpha f(\boldsymbol{x})),$$

(a) Predicted houses, $\beta = 0.999$ (b) Predicted rockets, $\beta = 0.001$ (c) Ambiguous images, $\beta = 0.5$

Figure 1: Given our trained classifier and generative world, we constrain β to infer images of a given confidence level. Qualitatively, the generated ambiguous images appear to be primarily either extremes drawn from the distribution (e.g., small values for rectangle height h) or images with aspect ratios in between those of prototypical houses and rockets as shown in Figure 2.

where α is a hyper-parameter to be set to determine the tolerance of an inaccurate $\tilde{\boldsymbol{p}}_x$: the lower the value of this hyperparameter, the more tolerant the sampling procedure.

In addition, rather than sampling directly from the data distribution, we sample in the latent factor space Z, which can be mapped to X via a deterministic reconstruction function $g : Z \to X$. We use the notation $\tilde{\boldsymbol{p}}_{\boldsymbol{z}} \doteq \tilde{\boldsymbol{p}}_{g(\boldsymbol{x})}$. We assume the latent space distribution $p(\boldsymbol{z})$ is much easier to sample from; for example, if our generative model is a GAN or VAE, this distribution is a unit Gaussian.

To summarize, given

$$\boldsymbol{z} \sim p(\boldsymbol{z}), \tag{1}$$

$$\boldsymbol{x} = g(\boldsymbol{z}), \tag{2}$$

$$\tilde{\boldsymbol{p}}_{\boldsymbol{z}} \sim \mathrm{Dir}(\alpha f(\boldsymbol{x})), \tag{3}$$

$$p(\boldsymbol{z} \,|\, \tilde{\boldsymbol{p}}_{\boldsymbol{z}} = \boldsymbol{p}) \propto p(\boldsymbol{z}) p(\tilde{\boldsymbol{p}}_{\boldsymbol{z}} = \boldsymbol{p} \,|\, \boldsymbol{z}), \tag{4}$$

where α is the hyper-parameter and $\boldsymbol{p}$ is the target prediction, we wish to sample from posterior $p(\boldsymbol{z} \,|\, \tilde{\boldsymbol{p}}_{\boldsymbol{z}} = \boldsymbol{p})$, and reconstruct $\boldsymbol{x}$ from $\boldsymbol{z}$ using $g(\cdot)$. We see that the posterior distribution is large only if the prior is large as well, limiting our sampled $\boldsymbol{z}$ (and therefore the reconstructed $\boldsymbol{x}$) to the generative distribution $p(\boldsymbol{x})$. In addition, we have $\lim_{\alpha \to \infty} ||f(g(\boldsymbol{z})) - \boldsymbol{p}|| = 0$ with probability 1, meaning that prediction of the samples will get closer to the target prediction as we increase α.

To sample from the posterior, we use the Metropolis-Hastings algorithm. We start from $\boldsymbol{z}^{(0)} \sim p(\boldsymbol{z})$ and then use a Gaussian proposal function centered on the current $\boldsymbol{z}$ with symmetric covariance matrix (i.e. kI for $k > 0$). To exploit parallelism of computer architectures, we start with N particles, and we sample for T time steps. In this process, we conservatively choose a relatively large T so that we sample beyond the requisite "burn-in" time. For each particle, the last sample $z^{(T)}$ of the trajectory is selected. Finally, we select $n \leq N$ such unweighted samples using resampling with replacement.

Experiments

We conduct experiments on two datasets: a synthetic image dataset in which images look either like a "house" or a "rocket" (see figure 2), and the MNIST dataset. For both datasets, we either have access to the true data distribution or learn a data distribution $p(\boldsymbol{z})$ over the latent space. We then sample n latent vectors $\{\boldsymbol{z}_i\}_{i=1}^n$ from the posterior distribution (Eqn 4) with certain known or learned data distribution (i.e. prior) and specified target prediction $\boldsymbol{p}$, and reconstruct $\{\boldsymbol{x}_i\}_{i=1}^n$ using the reconstruction function $g(\cdot)$.

To evaluate the quality of our sampled images, we compute the predicted probabilities: $\{\tilde{\boldsymbol{p}}_i = f(\boldsymbol{x}_i)\}_{i=1}^n$. We then calculate average deviation from target over class according to

$$\Delta \doteq \frac{1}{n} \frac{1}{|L|} \sum_{i=1}^{n} \sum_{l \in L} |\boldsymbol{p}_l - \tilde{\boldsymbol{p}}_{i,l}|, \tag{5}$$

where $\tilde{\boldsymbol{p}}_{i,l}$ is the predicted probability of $\boldsymbol{x}_i$ for class l.

Rockets or Houses: A Synthetic Geometric World

We construct a synthetic world where simple rules govern how an image of a triangle on top of a rectangle is drawn. Some of the images look like "houses" when a short triangle is put on top of a wide rectangle. Others look like "rockets"

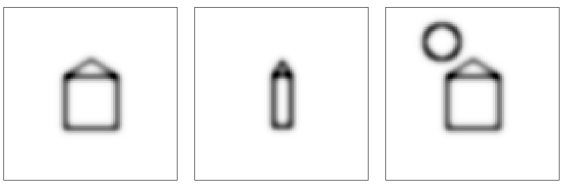


Figure 2: Left: the prototypical house image; middle: the prototypical rocket image; right: introduction of out of distribution circle geometry. Images have a Gaussian blur applied to them for ease of classification.

when a tall triangle is put on top of a slim rectangle. See figure 2 for prototypical images. Each image is rendered from three parameters: rectangle width w, rectangle height h, and triangle height t, which are generated from a mixture model conditioned on a latent variable c representing whether the image should be a house or a rocket. Specifically, to draw an image we follow the generative process:

$$
\begin{aligned}
c &\sim \mathrm{Ber}(0.5), \\
w|(c=0) &\sim \mathrm{No}_+(10, 5^2), \\
h|(c=0) &\sim \mathrm{No}_+(30, 10^2), \\
t|(c=0) &\sim \mathrm{No}_+(8, 2^2), \\
w|(c=1) &\sim \mathrm{No}_+(30, 10^2), \\
h|(c=1) &\sim \mathrm{No}_+(30, 10^2), \\
t|(c=1) &\sim \mathrm{No}_+(10, 2^2), \\
\boldsymbol{z} &\doteq [w, h, t]
\end{aligned}
$$

where $\mathrm{No}_+(\mu, \sigma^2)$ is the normal distribution truncated to the positive support. After rendering these images, we subsequently applied a Gaussian blur for for easier classification.

Using the synthetic images and the known groundtruth labels (c), we train a CNN-based classifier on 16,000 such images. Our classifier achieves 97.3% accuracy (94.7% precision, 97.4% recall). We then use our method to sample examples with target prediction $\boldsymbol{p} = [0.5, 0.5]$, with $\alpha = 10$. Note that in this binary case, the Dirichlet distribution reduces to the Beta distribution.

Figure 1 plots some of the images sampled from the posterior distribution with β constrained to produce images which confidently classify as houses ($\beta = 0.999$), images which confidently classify as rockets ($\beta = 0.001$), and images which classify as ambiguous ($\beta = 0.5$). Empirically, these images indeed appear to be houses, rockets, or ambiguous images. For generated houses, the mean classifier confidence was 94.9% (minimum 84.1%); for generated rockets, the mean classifier confidence was 94.9% (minimum 86.4%). The average generated image resulted in a classification confidence of 51.0%. This result is also confirmed in Figure 3, which visualizes the parameters of these ambiguous images rendered among the mixtures of two Gaussians. We compare our inference approach for selecting images from the three level sets to a set of randomly generated latents. We find many more ambiguous images through our inference approach than when directly sampling latents from the respective distributions.

Our method is not limited to test data from the distribution. We also demonstrate its potential for understanding behaviors of a trained classifier on different data distribution. To this end, we fix the classifier, trained on these "house" or "rocket" images, but we introduce a circle to the geometry rendering. A circle is randomly overlaid on an image as shown in Figure 2. We introduce the parameters circle radius r and circle center (x, y) into our generative process:

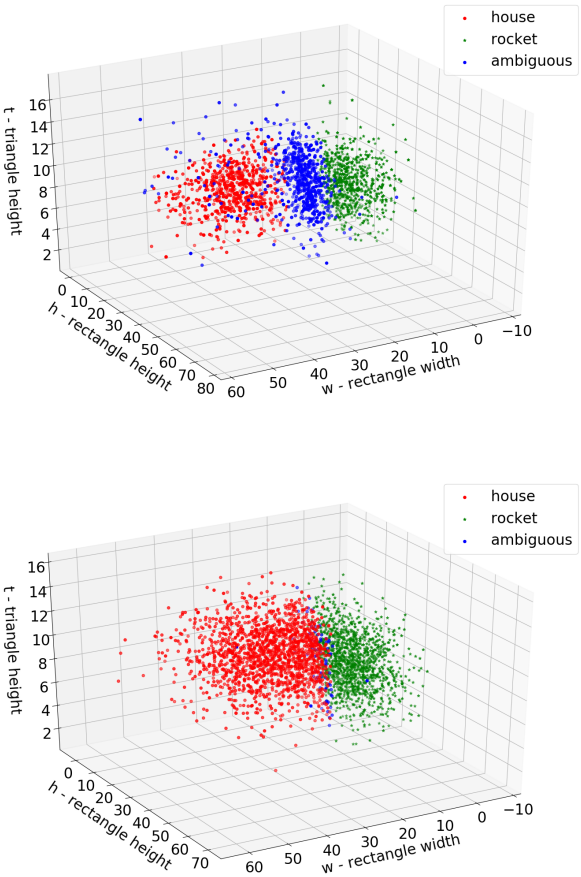


Figure 3: Above: three-dimensional latents discovered through sampling the posterior as predicted confidently houses, confidently rockets, or ambiguous. Below: 3000 latents generated directly by sampling the prior distributions. Ambiguous images have confidences ranging between 45% and 55% for each class. Indeed, we discover many more ambiguous examples through our targeted inference approach.

$$
\begin{aligned}
r &\sim \mathrm{No}_+(20, 10^2), \\
x &\sim \mathrm{U}(20, 40) \\
y &\sim \mathrm{U}(20, 40), \\
\boldsymbol{z} &\doteq [w, h, t, r, x, y]
\end{aligned}
$$

where w, h, and t are defined as before, and $\mathrm{U}(a, b)$ is a uniform distribution.

Using this new generative function, we again infer images from our three classes: those for which the classifier confidently predicts houses, those for which the classifier confidently predicts rockets, and those for which the classifier has low prediction confidence. We then evaluate how the presence of this out-of-distribution geometry affects classification predictions, and we find it has minor effects. For

	VAE	GAN
ambiguous	1.83%	5.93%
1vs7	6.22%	7.92%
8vs9	7.66%	5.31%

Table 1: Average deviation for different generative model and different prediction targets.

generated rocket images, average confidence in the rocket prediction falls 1.6% when circles are rendered; accuracy is unchanged. For generated house images, average confidence in the house prediction increases 1.1%; accuracy is likewise unchanged. For generated ambiguous images, average confidence in predictions increases 4.9% when circles are rendered. Curiously, when we infer ambiguous images with the circles rendered, the accuracy of the classifier is 15% higher than when the circles are not rendered.

MNIST

On the MNIST dataset, we learn to approximate the data distribution using both a variational autoencoder (VAE) and a generative adversarial network (GAN) as the source data distribution is not available. Both models are trained with a 5-dimensional latent space. Hence, we have two induced data distribution $p_{VAE}(\boldsymbol{x})$ and $p_{GAN}(\boldsymbol{x})$, derived from $p_{VAE}(\boldsymbol{z}) = p_{GAN}(\boldsymbol{z}) = \text{MVN}(\mathbf{0}, I) \in \mathbb{R}^5$. For each VAE and GAN prior, we studied three target predictions:

1. ambiguous, with $\boldsymbol{p}_l = 0.1 \ \ \forall l \in \{0, 1, ..., 9\}$;
2. 1vs7, with $\boldsymbol{p}_1 = \boldsymbol{p}_7 = 0.46$ and $\boldsymbol{p}_l = 0.01 \ \ \forall l \neq 1, 7$;
3. 8vs9, with $\boldsymbol{p}_8 = \boldsymbol{p}_9 = 0.46$ and $\boldsymbol{p}_l = 0.01 \ \ \forall l \neq 8, 9$.

Table 1 shows the average prediction deviation Δ as defined in Eqn 5. We can see that in general these models are indeed sampling data points around the specified target prediction. While they seem to have comparable performance, we note in the qualitative evaluations that sampling with a GAN prior will reduce much less valid samples than with a VAE prior.

VAE Visualization For qualitative evaluation, we visualize both the generated images and the latent space (reduced to 2 dimensions using t-SNE (Maaten and Hinton, 2008)).

The images sampled from the VAE prior for each class are shown in Figure 4. For the ambiguous target, we see that the depicted images are often the result of several digits overlaid on each other and lacking in intensity. For the 1vs7 target, we see that most samples are indeed "between" digit 1 and digit 7, in that the horizontal stroke is short but visible in most cases. For the 8vs9 target, the lower circle is in general smaller than the upper circle, which is also visually between digit 8 (which should have equally-sized circles) and digit 9 (in which the lower circle degenerates to a vertical stroke).

With the same VAE model, the sampled latent vectors for the three targets are plotted in Figure 4 after reduction to 2 dimensions using t-SNE. For Figure 4a, the lightly colored dots represent latent vectors sampled from the prior distribution (i.e. 5D unit Gaussian), color-coded according to the classifier prediction, and the dark green dots represent latent vectors sampled from the posterior (Eqn 4). For Figure 4b and 4c, the gray, blue and red dots represent latent vectors sampled from the prior distribution, and are color-coded according to classifier predictions. The dark green dots represent latent vectors sampled from the posterior.

We can see that the sampled latent vectors for the ambiguous case indeed lie at the intersections of several colored regions, though probably not between *all* regions, because it is hard for an image to look like all the 10 digits at the same time. In addition, they are also quite centrally located, indicating a high likelihood of drawing this digit according to VAE. For the latent vectors of 1vs7 and 8vs9, they are generally "in-distribution", although they do tend to occupy a smaller probability mass. This smaller mass is expected, as most of the highly likely images (including those seen in the training set) are indeed unambiguously a single digit. However, the sampled latent vectors are also not completely out of distribution, which is commonly believed to be the case for adversarial examples (Li et al., 2019).

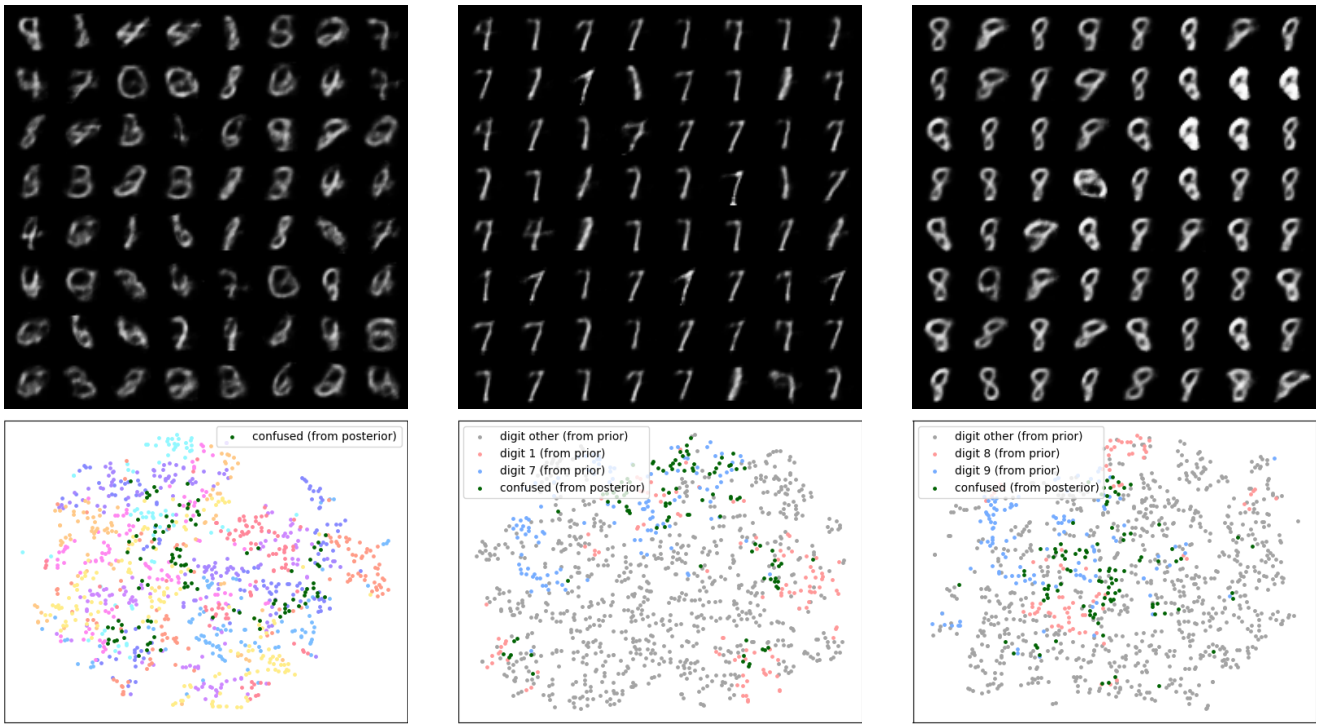
GAN Visualization For posterior sampling with a GAN as the learned generative model, the final resampling step (which is done with replacement) produces less than 10 distinct latent vector samples. Thus, we show the reconstructed images from all of them in Figure 5.

The small number of distinct samples in the GAN model results from the classifier being overly confident on most samples, an issue discussed in the Related Work section. This overconfidence makes it increasingly challenging to sample an image which produces ambiguous predictions from the generative model.

We see that in comparison with the VAE, the images reconstructed with the GAN model are much sharper. The GAN generator explicitly penalizes the production of blurry images that do not look like real images. Therefore, we believe that the sampler is using a different strategy: rather than sampling blurred images (which would have high likelihood under VAE but low likelihood under GAN), it samples images that look like hand-drawn digits but are not actual digits (e.g. last image of Figure 5a and second image of Figure 5c). One direct extension of this work is to try to use GAN model on some other handwritten datasets such as Omniglot (Lake, Salakhutdinov, and Tenenbaum, 2013) and Kuzushiji-MNIST (Clanuwat et al., 2018).

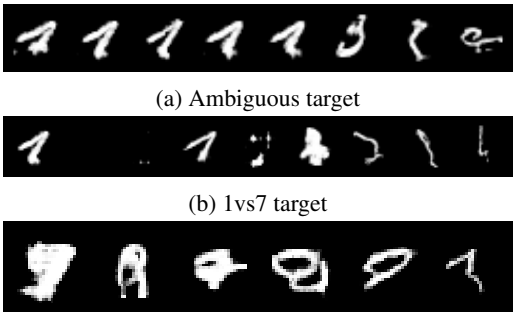
Conclusion and Future Work

In this paper, we propose a method to sample data that have known classification prediction values. Compared to directly performing unconstrained gradient descent from an initial data point, as is the case in most adversarial attack work, our method also regularizes the resulting examples such that the sampled data has high likelihood under some specified data distribution. To achieve this, we formulate the inference problem as a Bayesian one, and we use probabilis-



(a) Ambiguous target (b) 1vs7 target (c) 8vs9 target

Figure 4: Images inferred for specified prediction confidences sampled from the VAE's latent space, and visualizations of their presence in the latent space. Dark green dots represent selected ambigious samples. Left: images showing equal confidence in predictions for all digits $0, 1, \ldots, 9$. Middle: images showing equal confidence for 1 and 7 classifications. Right: images showing equal confidence for 8 and 9 classifications.



(a) Ambiguous target

(b) 1vs7 target

(c) 8vs9 target

Figure 5: Sampled images for three targets with GAN-learned data distribution.

tic programming to sample from the posterior. In addition to selecting "in-distribution" samples, our method operates globally: the initial starting point does not affect the final result after sufficiently many burn-in samples have been drawn and discarded. On both a synthetic dataset and the MNIST dataset, we demonstrate that our method can indeed sample data points close to the specified target while remaining likely under the given distribution. In addition, our technique may be useful for identifying overconfident networks: the lack of diversity of ambiguous samples with the MNIST GAN distribution indicates that the MNIST classifier may be overly confident on GAN-generated images.

There are several future directions for this work. First, MNIST is a relatively easy task with visually distinct classes. We will evaluate our technique on harder datasets such as CIFAR-10 or CIFAR-100 for which ambiguous labels (e.g. car vs truck for CIFAR-10 or different classes within one super-class in CIFAR-100) are more likely. Moreover, we can verify the effectiveness of confidence calibration (Guo et al., 2017) of the classifier using our model.

Our method could be also extended for in-distribution adversarial example discovery. For such discovery, we would require or learn a generative model for the entire dataset with examples for one class removed. Given this distribution, we would then sample examples from the posterior given the inference target of being highly confident as the missing class.

Finally, we can use this method to create neural network models with notions of ambiguity similar to those of a human. Specifically, we plan to iteratively train a classifier and augment the dataset to incorporate labeled examples drawn from the ambiguous sets. In the end, the classifier will imitate human prediction both in certain and in ambiguous cases and will hopefully make only human-like mistakes, which could be valuable for use in sensitive domains.

Acknowledgements

The authors would like to thank Alex Lew, Marco Cusumano-Towner, Christian Muise, and Hendrik Strobelt for fruitful discussions.

References

Athalye, A.; Engstrom, L.; Ilyas, A.; and Kwok, K. 2017. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*.

Athalye, A.; Carlini, N.; and Wagner, D. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*.

Balan, A. K.; Rathod, V.; Murphy, K. P.; and Welling, M. 2015. Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, 3438–3446.

Bingham, E.; Chen, J. P.; Jankowiak, M.; Obermeyer, F.; Pradhan, N.; Karaletsos, T.; Singh, R.; Szerlip, P.; Horsfall, P.; and Goodman, N. D. 2018. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, 1613–1622. JMLR. org.

Carlini, N., and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, 39–57. IEEE.

Carlini, N.; Athalye, A.; Papernot, N.; Brendel, W.; Rauber, J.; Tsipras, D.; Goodfellow, I.; and Madry, A. 2019. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*.

Clanuwat, T.; Bober-Irizar, M.; Kitamoto, A.; Lamb, A.; Yamamoto, K.; and Ha, D. 2018. Deep learning for classical japanese literature. *arXiv preprint arXiv:1812.01718*.

Cohen, J. M.; Rosenfeld, E.; and Kolter, J. Z. 2019. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*.

Corbett-Davies, S.; Pierson, E.; Feller, A.; Goel, S.; and Huq, A. 2017. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 797–806. ACM.

Cusumano-Towner, M. F.; Saad, F. A.; Lew, A. K.; and Mansinghka, V. K. 2019. Gen: A general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, 221–236. New York, NY, USA: ACM.

Doshi-Velez, F., and Kim, B. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Fremont, D. J.; Yue, X.; Dreossi, T.; Ghosh, S.; Sangiovanni-Vincentelli, A. L.; and Seshia, S. A. 2018. Scenic: Language-based scene generation. *arXiv preprint arXiv:1809.09310*.

Gal, Y., and Ghahramani, Z. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Graves, A. 2011. Practical variational inference for neural networks. In *Advances in neural information processing systems*, 2348–2356.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1321–1330. JMLR. org.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Lake, B. M.; Salakhutdinov, R. R.; and Tenenbaum, J. 2013. One-shot learning by inverting a compositional causal process. In *Advances in neural information processing systems*, 2526–2534.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 6402–6413.

Lecuyer, M.; Atlidakis, V.; Geambasu, R.; Hsu, D.; and Jana, S. 2019. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, 656–672. IEEE.

Lee, K.; Lee, H.; Lee, K.; and Shin, J. 2017. Training confidence-calibrated classifiers for detecting out-of-distribution samples.

Li, B.; Chen, C.; Wang, W.; and Carin, L. 2018. Second-order adversarial attack and certifiable robustness. *arXiv preprint arXiv:1809.03113*.

Li, Y.; Li, L.; Wang, L.; Zhang, T.; and Gong, B. 2019. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. *arXiv preprint arXiv:1905.00441*.

Lin, Z.; Khetan, A.; Fanti, G.; and Oh, S. 2018. Pacgan: The power of two samples in generative adversarial networks. In *Advances in Neural Information Processing Systems*, 1498–1507.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Nguyen, A.; Yosinski, J.; and Clune, J. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 427–436.

Rosenblatt, M. 1956. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics* 832–837.

Salvatier, J.; Wiecki, T. V.; and Fonnesbeck, C. 2016. Probabilistic programming in python using pymc3. *PeerJ Computer Science* 2:e55.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15(1):1929–1958.

Srivastava, A.; Valkov, L.; Russell, C.; Gutmann, M. U.; and Sutton, C. 2017. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, 3308–3318.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks.

Torrey, L., and Shavlik, J. 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI Global. 242–264.

Tran, D.; Kucukelbir, A.; Dieng, A. B.; Rudolph, M.; Liang, D.; and Blei, D. M. 2016. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*.