Bayes-TrEx: Model Transparency by Example

Serena Booth*, **Yilun Zhou***, **Ankit Shah**, **Julie Shah**
*Equal Contribution
CSAIL, Massachusetts Institute of Technology
{serenabooth, yilun, ajshah, julie_a_shah}@csail.mit.edu

Abstract

Post-hoc explanation methods are gaining popularity as tools for interpreting, understanding, and debugging neural networks. Most post-hoc methods explain decisions in response to individual inputs drawn from the test set. However, the test set often fails to include highly confident misclassifications and ambiguous examples. To address these challenges, we introduce BAYES-TREX for more flexible model inspection. It is a model- and data-generator-agnostic method for creating new distribution-conforming examples of known prediction confidence. BAYES-TREX can be used to find highly confident misclassifications; to visualize class boundaries through ambiguous examples; to understand novel-class extrapolation; and to expose neural network overconfidence. We demonstrate BAYES-TREX on CLEVR, MNIST, and Fashion-MNIST. Compared to inspecting test set examples, we show that BAYES-TREX enables more flexible holistic model analysis. Code: github.com/serenabooth/Bayes-TrEx.

1 Introduction

Debugging, interpreting, and understanding neural networks can be challenging (Odena et al. 2019; Lipton 2018; Doshi-Velez and Kim 2017). Existing interpretability methods include visualizing filters (Olah, Mordvintsev, and Schubert 2017; Zeiler and Fergus 2014), saliency maps (Zeiler and Fergus 2014; Simonyan, Vedaldi, and Zisserman 2013), input perturbations (Ribeiro, Singh, and Guestrin 2016; Lundberg and Lee 2017), prototype anchoring (Chen et al. 2019; Li et al. 2018), tracing with influence functions (Koh and Liang 2017), and concept quantification (Ghorbani, Wexler, and Kim 2019). While some post-hoc methods analyze intermediary network components such as convolutional layers (Bau et al. 2017; Olah, Mordvintsev, and Schubert 2017), most methods instead provide explanations to justify decisions in response to specified inputs. The inputs provided to these local explanation methods are typically drawn from the test set, but the test set may have poor representation of highly confident misclassifications and ambiguous examples. These limitations can make it challenging to extract meaningful insights with only test set inputs. Finding and analyzing varied inputs that invoke the gamut of model behaviours would improve *transparency by example*.

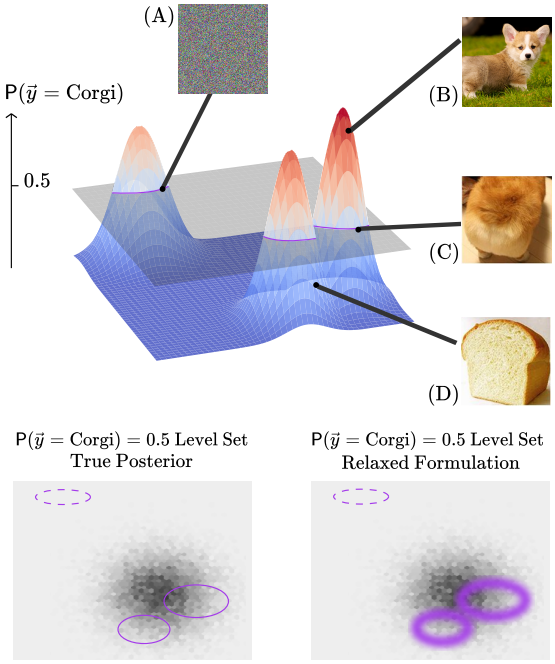
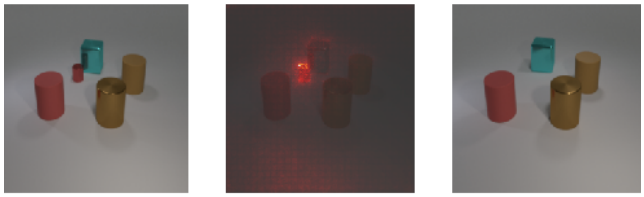


Figure 1: Given a Corgi/Bread classifier and a generative model for the data, we generate *prediction level sets*, or sets of examples of a target prediction confidence. Adversarial approaches perturb a given image to the target prediction confidence (e.g., $P_{\text{Corgi}} = P_{\text{Bread}} = 0.5$). Without image prior regularization, such examples are "out-of-distribution" as shown in (A). BAYES-TREX takes a specified data distribution to sample "in-distribution," or natural, examples in the target prediction level set (e.g., likely Corgi (B), likely Bread (D), or ambiguous between Corgi and Bread (C)). Bottom left: the classifier level set of ($P_{\text{Corgi}} = P_{\text{Bread}} = 0.5$) overlaid on the data distribution heatmap. Example (A) would not be sampled because its likelihood is low under the generative model, while example (C) would be sampled. Bottom right: As sampling directly from the true posterior is infeasible, we relax the formulation by widening the level set. By specifying different data distributions, we can uncover inputs that invoke various model behaviors to improve model transparency by example.



$P_{1\ \text{Sphere}} = 97.1\%$ Saliency Map $P_{1\ \text{Sphere}} = 0.1\%$

Figure 2: *Inspecting a model.* We ask BAYES-TREX to find a CLEVR scene that contains no spheres yet is misclassified as containing a sphere. The generated example (left) is composed of only cylinders and cubes, but the classifier is 97.1% confident this scene contains one sphere. Using SmoothGrad (Smilkov et al. 2017), the saliency map highlights the small red cylinder as the object that is confused for a sphere. When we remove it, the classifier's confidence that the scene contains one sphere drops to 0.1%.

Expanding the test set without curating new labeled data is impossible, but the underlying data distribution can be represented by generative models—whether rendered, learned, or procedural. BAYES-TREX is a tool for sampling inputs at a known prediction confidence: for example, $P_{\text{Corgi}} = 0.7, P_{\text{Bread}} = 0.3$ for a Corgi/Bread classifier (Fig. 1). We call the set of all input examples that meet this prediction confidence the $\mathbf{p}$-level set. Given a data distribution and a classifier, BAYES-TREX generates level-set examples from the posterior of a hierarchical Bayesian model by applying Markov Chain Monte-Carlo (MCMC) inference methods. We show BAYES-TREX being used with a rendered scene graphs (CLEVR (Johnson et al. 2017)), and with learned generative models (GANs and VAEs) for organic datasets MNIST (LeCun and Cortes 2010) and Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017).

BAYES-TREX is a tool for holistic model evaluation and can be used to aid transparency by example across several contexts. By specifying an ambiguous prediction target, BAYES-TREX can generate examples to visualize class boundaries. By restricting the generator to not produce certain classes, BAYES-TREX can generate examples which are incorrectly classified with high confidence. By introducing novel classes for generation, BAYES-TREX can help model designers anticipate how their models will behave in new scenarios. Further, BAYES-TREX examples can be used as input to existing local explanation techniques, like saliency maps (Fig. 2). Lastly, we show how BAYES-TREX can be used to expose network overconfidence, as we demonstrate on the domain-adapted model ADDA (Tzeng et al. 2017).

The main current alternative to BAYES-TREX is to inspect a model by using test set examples. As a baseline comparison, we search for highly confident misclassifications and ambiguous examples in the CLEVR, MNIST, and Fashion-MNIST test sets. We find that very few test set examples provide these specific insights into model behavior; in contrast, BAYES-TREX consistently finds more highly confident misclassifications and ambiguous examples, which enables more flexible model inspection.

2 Related Work

2.1 Interpreting Neural Networks

A typical approach for interpreting a neural network is to view the model's activation-maximizing 'filters' (Olah, Mordvintsev, and Schubert 2017; Erhan et al. 2009). Without regularization, these filters are unnatural images; to address this, Nguyen et al. used a generative adversarial network as a prior to ensure the generated filters appear realistic, balancing activation maximization with realism (Nguyen et al. 2016). While these activation-maximizing filters can help diagnose individual units in a neural network, understanding pattern-matching between inputs and filters remains challenging (Olah, Mordvintsev, and Schubert 2017; Nguyen, Yosinski, and Clune 2019). Instead of finding optimized images which maximize neuron activations, BAYES-TREX finds realistic, in-distribution inputs which invoke a specified model response—including arbitrary prediction confidence targets such as ambiguity between classes.

Unlike activation-maximizing filters, saliency maps are a *local* or *per-example* method: they let us inspect model behavior given a specific example by projecting feature activations to input pixel space to highlight regions of interest (Zeiler and Fergus 2014; Simonyan, Vedaldi, and Zisserman 2013). Examples are typically drawn from test set failures; however, the test set may only sparsely invoke some model responses. While saliency maps are imperfect and can appear reasonable even when output from an untrained network (Kindermans et al. 2019; Adebayo et al. 2018), they have been used successfully for model diagnosis (Zeiler and Fergus 2014) and are now ubiquitous tools for interpreting neural networks. That said, model explanations require a two-stage pipeline. First, we need to find interesting examples, leveraging both train and test data distributions. Then, we can invoke interpretability methods to explain these examples (Fig. 2, Appendix L). To our knowledge, BAYES-TREX is the first work focused on finding interesting inputs for inspection with local explanation methods.

2.2 Adversarial Examples for Neural Networks

One BAYES-TREX use case is to uncover high-confidence classification failures in the data distribution. This idea is related to, but different from, natural adversarial attacks (Zhao, Dua, and Singh 2018). While most adversarial attacks rely on injecting carefully crafted high-frequency information to mislead a trained model (Szegedy et al. 2013; Nguyen, Yosinski, and Clune 2015; Goodfellow, Shlens, and Szegedy 2014; Carlini and Wagner 2017), Zhao et al. (2018) proposed a method to find *natural* adversarial examples which are near to a specific example but cause misclassification. To preserve naturalness, the method optimizes within the latent space of a GAN. This method is local because it starts with a specific input. In contrast, BAYES-TREX is not restricted to finding examples near a specific test input and finds examples by using the global data distribution.

2.3 Confidence in Neural Networks

BAYES-TREX draws examples from confidence-based level sets in neural networks. Guo et al. showed that many neural networks are overconfident, with incorrect predictions often having high confidence (2017). While many approaches aim to address this network overconfidence problem (Thulasidasan et al. 2019; Lee et al. 2017; Gal and Ghahramani 2016; Blundell et al. 2015), our work is complementary to these efforts. Rather than altering the confidence of a neural network, BAYES-TREX instead infers examples of a particular confidence. If the model is overconfident, BAYES-TREX may return few, if any, samples with ambiguous predictions. Meanwhile, BAYES-TREX may find many misclassifications with high confidence. As such, we can use BAYES-TREX to assist in the diagnosis of overconfident networks: In our experiments (Section 4.6), we find that the popular domain adaptation technique ADDA produces a more overconfident model than a baseline non-adapted model.

3 Methodology

Given a classifier $f : X \to \Delta_K$ which maps a data point to the probability simplex of K classes, the goal is to find an input $\mathbf{x} \in X$ in a known distribution $p(\mathbf{x})$ such that $f(\mathbf{x}) = \mathbf{p}$ for some particular prediction confidence $\mathbf{p} \in \Delta_K$. We consider the inference problem of sampling from the posterior

$$p(\mathbf{x}|f(\mathbf{x}) = \mathbf{p}) \propto p(\mathbf{x})\, p(f(\mathbf{x}) = \mathbf{p}|\mathbf{x}). \tag{1}$$

A common approach to sampling from the posterior is to use Markov Chain Monte-Carlo (MCMC) methods (Neal et al. 2011). However, when the measure of the level set $\{\mathbf{x} : f(\mathbf{x}) = \mathbf{p}\}$ is small or even zero, sampling directly from this posterior using MCMC is infeasible: the posterior being zero everywhere outside of the level set means that it is unlikely for a Random-Walk Metropolis sampler to land on $\mathbf{x}$ with non-zero posterior, and the gradient being zero everywhere outside of the level set means that a Hamiltonian Monte Carlo sampler does not have the necessary guidance toward the level set either.

To enable inference, we instead solve a relaxed version of the problem, which we can anneal to be arbitrarily close to the original target. We relax the formulation by accepting $\mathbf{x}$ when $f(\mathbf{x})$ is close to the target $\mathbf{p}$ (Fig. 1). Let $\mathbf{p} = [p_1, \ldots, p_K]^T$. Let $\mathbf{u}$ be a random vector where $\mathbf{u} = [u_1, \ldots, u_K]^T$. We sample u_i according to:

$$u_i|\mathbf{x} \sim \mathcal{N}\left(f(\mathbf{x})_i, \sigma^2\right), \qquad u_i^* = p_i \tag{2}$$

where σ is a hyper-parameter determining the extent of level-set relaxation (see Figure 1).

Instead of directly sampling from Eqn. 1, we can now sample from the new posterior:

$$p(\mathbf{x}|\mathbf{u} = \mathbf{u}^*) \propto p(\mathbf{x})p(\mathbf{u} = \mathbf{u}^*|\mathbf{x}). \tag{3}$$

Under expectation, when the sampled data has high probability, the classifier prediction matches $\mathbf{p}$.

Since $\mathbf{u}$ is sampled from a normal distribution, this formulation is asymptotically exact. Formally:

$$\lim_{\sigma \to 0} p(\mathbf{x}|\mathbf{u} = \mathbf{u}^*) = p(\mathbf{x}|f(\mathbf{x}) = \mathbf{p}) \tag{4}$$

As σ goes to 0, the approximate posterior distribution approaches the true posterior distribution of interest.

While the formulation in Equation 2 is applicable to arbitrary confidence assignments, it requires the number of auxiliary variables in $\mathbf{u}$ to be equal to the number of classes, posing scalability issues for large numbers of classes. We consider two specific instantiations for important BAYES-TREX uses: 1. Sampling high confidence examples, and 2. Sampling ambiguous examples. These use cases correspond to finding highly confident misclassified examples, novel class extrapolation examples, and class boundary examples. For the instantiations in Equations 5–7, the dimensionality of $\mathbf{u}$ is reduced to characterize the classes of interest.

1. High confidence examples: $p_i = 1, p_{\neg i} = 0$. The classifier should be as confident in its class i prediction as possible—irrespective of the correctness of the classification. For this case:

$$u|\mathbf{x} \sim \mathcal{N}\left(f(\mathbf{x})_i, \sigma^2\right), \qquad u^* = 1. \tag{5}$$

2. Ambiguous examples: $p_i = p_j = 0.5, p_{\neg i,j} = 0$. The classifier should be ambiguous between class i and class j, while not predicting any other classes. Consider $\mathbf{u} = [u_1, u_2]^T$ for this case:

$$u_1|\mathbf{x} \sim \mathcal{N}\left(|f(\mathbf{x})_i - f(\mathbf{x})_j|, \sigma_1^2\right) \tag{6}$$

$$u_2|\mathbf{x} \sim \mathcal{N}(\min(f(\mathbf{x})_i, f(\mathbf{x})_j) - \max_{k \neq i,j} f(\mathbf{x})_k, \sigma_2^2) \tag{7}$$

$$u_1^* = 0, u_2^* = 0.5$$

σ_1 and σ_2 are hyperparameters.

Rather than sampling directly from the data distribution, we sample in the latent factor space of a generative model, Z. Z is mapped to X through a deterministic reconstruction function $g : Z \to X$.

To summarize, given

$$\mathbf{x} = g(\mathbf{z}), \tag{8}$$

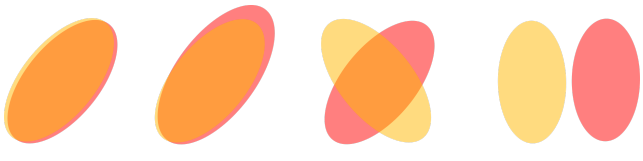
$$\mathbf{u}|\mathbf{z} \sim \mathcal{N}(f(\mathbf{x}), \sigma^2), \tag{9}$$

$$p(\mathbf{z}|\mathbf{u} = \mathbf{u}^*) \propto p(\mathbf{z})p(\mathbf{u} = \mathbf{u}^*|\mathbf{z}), \tag{10}$$

we wish to sample $\mathbf{z}$ according to Eqn. 10 and reconstruct the example $\mathbf{x} = g(\mathbf{z})$ for model inspection.

3.1 Inference Details

While there are many inference methods which may be used to sample from the relaxed posterior, we implement in particular two Markov Chain Monte Carlo methods: Random-Walk Metropolis and Hamiltonian Monte Carlo (Neal et al. 2011). In our experiments, we apply Random-Walk Metropolis when the prediction is not differentiable with respect to $\mathbf{z}$ (for example, when rendering is non-differentiable). When a gradient is available, we instead apply Hamiltonian Monte Carlo. For the Hamiltonian Monte Carlo method, we use the No-U-Turn sampler (Hoffman and Gelman 2014; Neal et al. 2011) and the probabilistic programming language Pyro for implementation (Bingham et al. 2018). All hyperparameters σ are arbitrarily chosen to be 0.05 for all experiments.



(a) $\mathbb{P}_C = \mathbb{P}_G$ (b) $\mathbb{P}_G \subsetneq \mathbb{P}_C$ (c) $\mathbb{P}_G \cap \mathbb{P}_C \neq \mathbb{P}_G \neq \mathbb{P}_C$ (d) $\mathbb{P}_G \cap \mathbb{P}_C = \emptyset$

Figure 3: Relations of classifier training data ($\mathbb{P}_C$, red) and generator training data ($\mathbb{P}_G$, yellow). (a) $\mathbb{P}_C$ and $\mathbb{P}_G$ are equal. (b) $\mathbb{P}_G$ is a subset of $\mathbb{P}_C$. (c) $\mathbb{P}_G$ and $\mathbb{P}_C$ overlap. (d) $\mathbb{P}_C$ and $\mathbb{P}_G$ are disjoint.

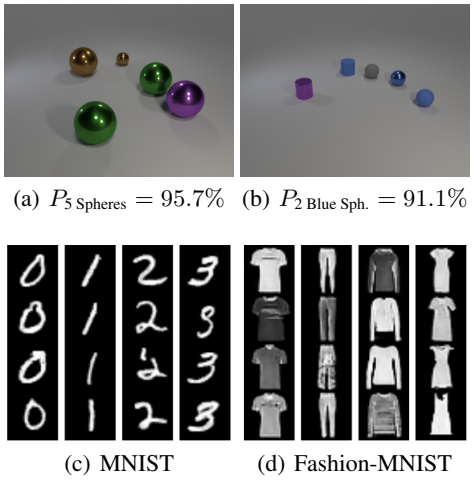
Selecting appropriate stopping criteria for MCMC algorithms is an open problem. State-of-the-art approaches require a gold standard inference algorithm (Cusumano-Towner and Mansinghka 2017) or specific properties of the posterior distribution, such as log-concavity (Gorham and Mackey 2015). As neither of these requirements are met for our general use cases, we select stopping criteria based on heuristic performance and cost of compute. As CLEVR scenes require GPU-intensive rendering, we select a stopping critieria of 500 samples for all but one experiment (Appendix D). For MNIST and Fashion-MNIST where samples are cheaper to generate, we select a stopping criteria of 2000 samples. Empirically, we found each CLEVR sample evaluation took 3.75 seconds, MNIST samples took 1.18 seconds, and Fashion-MNIST samples took 1.96 seconds on a single NVIDIA GeForce 1080 GPU.

4 Experiments

BAYES-TREX enables the evaluation of a classifier on a target generative distribution $\mathbb{P}_G$ irrespective of the distribution of the classifier training set $\mathbb{P}_C$. We demonstrate the versatility of BAYES-TREX on four relationships between $\mathbb{P}_G$ and $\mathbb{P}_C$ (Fig. 3). In Section 4.3, we find high confidence examples by considering $\mathbb{P}_C = \mathbb{P}_G$ (Fig. 3(a)). In Section 4.4, we consider $\mathbb{P}_G$ with narrower support than $\mathbb{P}_C$ (Fig. 3(b)), where $\mathbb{P}_G$ cannot generate data from a particular class. In this case, high-confidence samples—as judged by the classifier C—with P_G as the generative distribution reveals misclassified examples. In Section 4.5 and 4.6, we analyze the classifier C for out-of-distribution extrapolation and domain adaptation behaviours with overlapping or disjoint supports of $\mathbb{P}_C$ and $\mathbb{P}_G$ (Fig. 3(c) and (d)). Representative results are in the main text; further results are in the appendix.

4.1 Datasets

We evaluate BAYES-TREX on rendered data (CLEVR) and organic data (MNIST and Fashion-MNIST). In all CLEVR experiments, we use the pre-trained classifier distributed by the original authors (Johnson et al. 2017). Since CLEVR rendering is not differentiable, we use Random-Walk Metropolis for sampling. For probabilistic programming, we use a custom prior, consisting of a Gaussian proposal for the continuous variables (e.g., x-position) and categorical proposal for the discrete variables (e.g., color), with a high probability for the current value and uniform low probabilities for other values. For MNIST and Fashion-MNIST experiments, we train the classifiers using the architectures



(a) $P_{\text{5 Spheres}} = 95.7\%$ (b) $P_{\text{2 Blue Sph.}} = 91.1\%$

(c) MNIST (d) Fashion-MNIST

Figure 4: High-confidence samples. (a, b) CLEVR, with target class and predicted confidence. (c) MNIST Digits 0-3. (d) Fashion-MNIST classes, from left to right: T-shirt, trousers, pullover and dress.

and training details described in Appendix B. For domain adaptation analysis, we use the ADDA model and training code provided by the authors (Tzeng et al. 2017).

For CLEVR, we render scene graphs, and the renderer serves as the data generator. For (Fashion-)MNIST, we train and use both VAE and GAN models as the data generators. The latent distribution for these GANs and VAEs—$p(\mathbf{z})$—is a unit Gaussian. VAEs and GANs are known to be imperfect representations of underlying data distributions (Arora and Zhang 2017). To evaluate how well these generative models reproduce the training distributions, we report Fréchet Inception Distance (Heusel et al. 2017); see Appendix C.

4.2 Quantitative Evaluation

To evaluate the efficacy of BAYES-TREX, we compare the confidence scores of BAYES-TREX-generated examples to the specified prediction target. This analysis confirms that the inferred samples have predicted confidence closely matching the specified targets, indicating that the MCMC methods used by BAYES-TREX are successful for the tested domains and scenarios. A subset of these results are presented in Table 1. The full evaluation is in Appendix D.

4.3 High, Ambiguous, and Graded Confidence

As a first test, we evaluate BAYES-TREX by finding highly confident examples. Figure 4 depicts samples on all three datasets; additional high confidence examples are in Appendix E. Next, we find examples where the neural network is ambiguous between two classes. We use a VAE generative model for (Fashion-)MNIST. Figure 5 shows a matrix of ambiguous examples from each pair of classes (e.g. 0v1, 0v2, ..., 8v9). Note the examples presented are ambiguous from the classifier's perspective, and some may be readily classified by a human. For example, the 3v7 example is a 7, and Bag vs Coat example is a bag. As could be expected for

Table 1: Mean and standard deviation of the predicted confidence of the samples over 10 trials. Prediction confidence is for the target class, or two target classes in binary ambiguous and graded confidence cases. Appendix D presents the comprehensive extension of this table for all experiments.

Evaluation	Dataset	Target	Prediction Confidence
High Conf.	MNIST	$P_4 = 1$	1.00 ± 0.01
	Fashion	$P_{\text{Coat}} = 1$	0.98 ± 0.02
	CLEVR	$P_{\text{2 Blue Spheres}} = 1$	0.89 ± 0.25
Ambiguous	MNIST	$P_1 = 0.5, P_7 = 0.5$	$0.49 \pm 0.02, 0.49 \pm 0.03$
	Fashion	$P_{\text{T-shirt}}, P_{\text{Dress}}$	$0.48 \pm 0.02, 0.48 \pm 0.02$
Natural Adv.	MNIST	$P_8 = 1$	0.98 ± 0.02
	Fashion	$P_{\text{Bag}} = 1$	0.97 ± 0.03
	CLEVR	$P_{\text{1 Cube}} = 1$	0.93 ± 0.06
Extrapolation	MNIST	$P_6 = 1$	1.00 ± 0.01
	Fashion	$P_{\text{Sandal}} = 1$	1.00 ± 0.01
	CLEVR	$P_{\text{1 Cylinder}} = 1$	0.96 ± 0.03
Domain Adapt	MNIST	$P_5 = 1$	1.00 ± 0.01

a well performing classifier, not all pairs result in successful sampling: for example, we were unable to find an ambiguous example with equal prediction confidence between the dissimilar classes 7 and 0. While BAYES-TREX did not find an example for every class pairing, it was more substantially effective at this task than simply using the test set. BAYES-TREX found ambiguous examples for 38 MNIST class combinations, while the test set had examples for only 10 class combinations. For Fashion-MNIST, BAYES-TREX found examples for 28 class combinations, while the test set had 12 class combinations. Appendix F contains more examples and a latent space visualization of class boundaries. In addition to ambiguous examples, BAYES-TREX can also sample from level sets which interpolate between classes: Appendix G contains a complete interpolation between two classes. Finally, BAYES-TREX can find targets which are ambiguous between more than two classes (Appendix F).

4.4 High-Confidence Failures

BAYES-TREX can also find highly confident classification failures. With neural networks being increasingly used for high-stakes decision making, having an accurate measure of uncertainty is particularly important (Mitchell et al. 2019), making high-confidence failures especially harmful. We can expose class-c high-confidence classification failures by using BAYES-TREX with a generator distribution that does not include class-c examples. For this case, the generator distribution support is a subset of the classifier's support (Fig. 3(b)). For CLEVR, we revoke the generative model's ability to produce objects of a target class and sample high confidence images for that target (Fig. 6(a) and 6(b)). In Fig. 6(a), the classifier is highly confident that there is one cube. As confirmed by the saliency map and re-rendering (Appendix L), the classifier mistakes the shiny red cylinder for a cube. Collating such examples can enable data augmentation to increase network reliability (Fremont et al. 2019).

For (Fashion-)MNIST, for each class c, we train a GAN on the dataset with class-c removed. Fig. 6(c) and 6(d) depict high-confidence misclassifications for digits 0-4 in

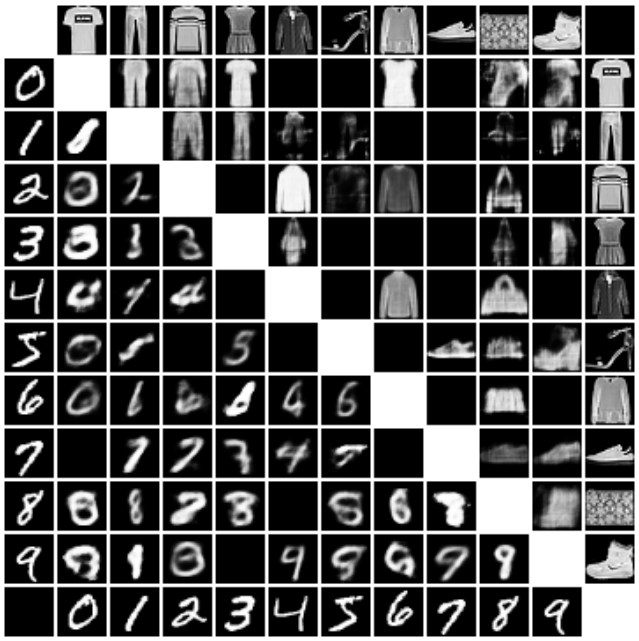
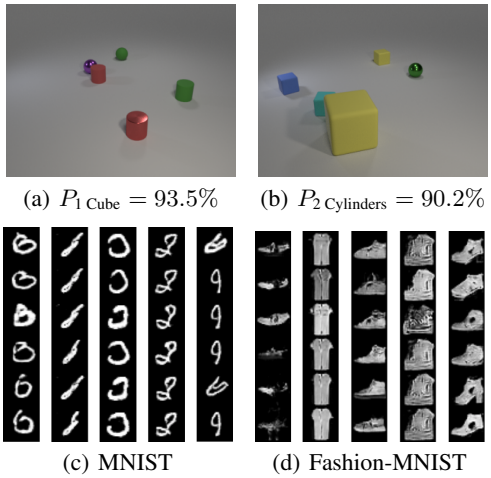


Figure 5: Each entry of the matrix is an ambiguous MNIST or Fashion-MNIST example for the classes on its row and column. Blacked-out cells indicate sampling failures.

MNIST and sandal, shirt, sneaker, bag, and ankle boot in Fashion-MNIST, respectively. We can assess how well human-aligned a classifier is by evaluating these examples. For example, for MNIST, some thin 8s are classified as 1s and particular styles of 6s and 9s are classified as 4s. These results seem intuitive, as a human might make these same mistakes. Likewise, for Fashion-MNIST, most adversarial examples come from semantically similar classes, e.g. sneaker $\leftrightarrow$ ankle boot. Less intuitively, however, chunky shoes are likely to be classified as bags. Interestingly, this misclassification propensity for chunky shoes as bags is not evident from test set evaluations. As shown in the confusion matrix (Appendix A, Fig. 9), no examples of ankle boots from the test set are misclassified as bags. Further, of the 17 highly confident misclassified sneakers, 12 are misclassified as ankle boots while 1 is labeled as a bag. In fact, this particular image is *mislabeled*: it is a bag, not a pair of sneakers. This incorrect model behaviour is not manifested in the test set, yet BAYES-TREX found it. Additional experiments on high confidence misclassification are in Appendix I.

4.5 Novel Class Extrapolation

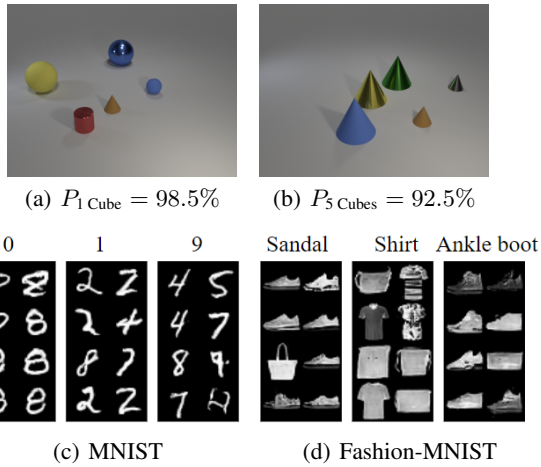
BAYES-TREX can also be used to understand model extrapolation behaviours on novel classes. Consider an autonomous vehicle. This vehicle might learn to safely operate around pedestrians, cyclists, and cars. But can we predict how the vehicle will behave when it encounters a novel class, like a tandem bicycle? To explore this use case, we constrain the generator such that it can produce examples of a novel class not present in the classifier's training data but cannot produce examples of a target class (Fig. 3(c, d)). For



(a) $P_{\text{1 Cube}} = 93.5\%$ (b) $P_{\text{2 Cylinders}} = 90.2\%$

(c) MNIST (d) Fashion-MNIST

Figure 6: High confidence classification failures, using a generator which is unable to produce examples of a target class. (a): CLEVR, 1 Cube. Note that no cube is present in the sample. (b): CLEVR, 2 Cylinders—again, containing no cylinders. (c) MNIST failures for digits 0-4. 0s are composed of 6s; 1s of 8s; 2s of 0s, and so on. (d) Fashion-MNIST failures for sandal, shirt, sneaker, bag, and ankle boot.



(a) $P_{\text{1 Cube}} = 98.5\%$ (b) $P_{\text{5 Cubes}} = 92.5\%$

(c) MNIST (d) Fashion-MNIST

Figure 7: Novel class extrapolation examples. (a, b): For CLEVR, we introduce a new shape–a cone–to the generative model, revoke the generative model's ability to produce target class objects, and infer high-confidence examples. The cones are mistaken for cubes. (c, d): For (Fashion-)MNIST, we train classifiers on subsets of the data (digits 0, 1, 3, 6, 9 for MNIST and Pullover, Dress, Sandal, Shirt, and Ankle boot for Fashion-MNIST), and train the data generators with the excluded data. The incorrect predicted class label is shown above the images. Average prediction confidence is ≈99%. Additional details in Appendix J.

a CLEVR instance, we equip the generator with the ability to produce a novel object—a cone—but revoke its ability to generate cubes. We observe that the classifier confidently mistakes cones for cubes (Fig. 7(a), 7(b); App. L). When an MNIST classifier is trained without digits 7 and 8 in its training data, the classifier confidently mistakes these for digits 0, 1, and 9 (Fig. 7(c)). A Fashion-MNIST classifier confuses novel sneakers for sandals and ankle boots. While confusing sandals and ankle boots is a seemingly reasonable extrapolation behavior, the classifier also confidently mistakes different bags as sandals, shirts, and ankle boots (Fig. 7(d)). Additional experiments in Appendix J.

4.6 Domain Adaptation

BAYES-TREX can also be used to analyze domain adaptation. Specifically, consider the SVHN (Netzer et al. 2011) → MNIST domain adaptation problem. We train two classifiers, a baseline classifier on labeled SVHN data only, and the adversarial discriminative domain adaptation (ADDA) classifier (Tzeng et al. 2017) on labeled SVHN data and unlabeled MNIST data. Indeed, domain adaptation improves classification accuracy: the baseline classifier achieves 61% accuracy on MNIST while the ADDA classifier achieves 71%, a clear performance improvement.

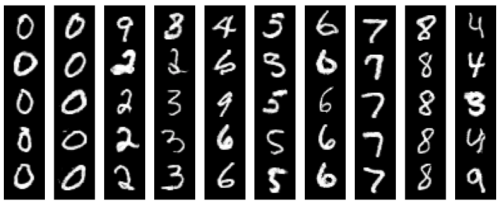
But is this the whole story? As misclassified high confidence examples are most harmful (Mitchell et al. 2019), we use BAYES-TREX to generate high confidence examples for both classifiers using an MNIST GAN (Fig. 8). Then, we hand re-label these examples and compute the accuracy. We find the baseline model achieves higher accuracy on these high confidence examples (80% vs 71%; see Supp. Table 11). This fine-grained analysis with BAYES-TREX suggests the ADDA model is more overconfident than the baseline. See Appendix K for experiment protocol and analysis.

4.7 Ambiguous Examples and Overconfidence

In Section 4.3 and Figure 5, we show how BAYES-TREX can infer ambiguous examples for (Fashion-)MNIST when using VAEs as the generative model. However, inference fails when using GANs. The FID scores show that the (Fashion-)MNIST GANs are better generators than their VAE counterparts (see Appendix C), and we observe the GAN-generated images to be generally sharper and more visually realistic. We assert the failure to infer ambiguous examples with a GAN is due to the network being consistently confident when evaluating realistic, sharp images. We experimentally verify this: when we explicitly train a classifier to have ambiguous confidence using a KL-divergence loss, using a GAN to sample ambiguous targets succeeds. This experiment is described in detail in Appendix H.

BAYES-TREX is also unable to generate ambiguous examples for CLEVR. While the (Fashion-)MNIST networks are high performing, with accuracies of ≈99%, CLEVR accuracy is markedly worse, ≈60% (Johnson et al. 2017). By definition, a high-performing network is not necessarily overconfident, even when highly confident on all examples (Guo et al. 2017). While the (Fashion-)MNIST networks may not be overconfident according to this definition, BAYES-TREX's inability to sample ambiguous CLEVR examples is likely due to this overconfidence problem. Indeed, this problem has previously been observed in CLEVR-style settings (Kim, Ricci, and Serre 2018).

(a) Baseline examples



(b) ADDA examples

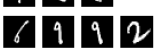
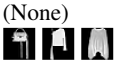
Figure 8: Highly confident examples for each class (0 to 9) of the baseline model and ADDA model. A manual labeling reveals *more* misclassified high-confidence examples with ADDA than with the baseline (Appendix K).

4.8 Baseline Comparison

Standard practice for model evaluation requires inspection of test set images. While inspecting test set examples is not an apples-to-apples comparison for all BAYES-TREX use cases, this can be used for evaluating ambiguous examples and classification failures. For ambiguous examples, we find examples in the (Fashion-)MNIST datasets where the classifier has confidence in $[40\%, 60\%]$ for two classes. Out of 10,000 MNIST test examples, we find only 12 ambiguous examples across 10 class combinations. Out of 10,000 Fashion-MNIST test examples, we find 162 ambiguous examples across 12 class combinations. BAYES-TREX found ambiguous examples for 29 class combinations which were not represented in the MNIST test set and 18 class combinations not represented in the Fashion-MNIST test set. BAYES-TREX consistently finds examples for most class combinations (Appendix D). The full test image matrix is shown in Appendix A, Figure 10.

We also evaluate highly confident test set misclassifications ($\geq 85\%$). For CLEVR, out of $15,000$ test images, the baseline discovered between 0 and 15 examples for each adversarial target (Appendix A). The baseline found no examples where the classifier confidently misclassified scenes as containing 2 cylinders, but Bayes-TrEx was successful in this task (Appendix D). For MNIST, out of the $10,000$ test images, 84 were confidently misclassified. However, the majority (60/84) of these misclassified examples have incorrect ground truth labels; see Table 2 and Appendix A. Fashion-MNIST had a total of 802 misclassifications. We hand-inspected 10 such misclassifications for each class (except 'trousers,' which had 3 total misclassifications). Of these inspected examples, we again found the majority (52/93) to be due to incorrect ground truth labels. Identifying mislabeled examples may be useful for correcting the dataset, but is not useful for our task of model inspection.

Table 2: High confidence misclassifications from the test set. The majority of these test set misclassifications are due to incorrect ground truth labels, not classifier failures. Full table of all (Fashion-)MNIST classes in Appendix A.

Class	Cause	Images
0	Misclassified	
	Mislabeled	
1	Misclassified	
	Mislabeled	
2	Misclassified	(None)
	Mislabeled	
Trouser	Misclassified	(None)
	Mislabeled	
Bag	Misclassified	
	Mislabeled	

5 Discussion

By specifying and varying the underlying distributions, BAYES-TREX can generate examples beyond the scope of the test set to provide more flexible transparency by example. Still, extracting insights from individual examples remains challenging, even with the use of downstream local explanation methods like saliency maps (Figure 2). To extract further insights and make BAYES-TREX easier for model designers to use, future work should additionally cluster and visualize trends in these generated examples, and estimate overall coverage of the level set.

While BAYES-TREX is in principle agnostic to choice of generative model, learned models like GANs and VAEs are currently the most widely used options for organic data. However, GANs and VAEs are imperfect representations of the underlying data distributions (Arora and Zhang 2017). These limitations of learned generative models affect the diversity and quality of examples discovered when using BAYES-TREX with organic data. Additionally, GANs and VAEs are computationally expensive to train. MCMC sampling methods are likewise resource intensive. While some learned generative models show promise for sample quality and diversity (Song and Ermon 2019), BAYES-TREX would also benefit from faster generative model training.

Finally, the current BAYES-TREX formulation only works for classification problems without dependencies among outputs. Nonetheless, the underlying empirical approach to transparency by example would be useful for applications which require accounting for such dependencies—for example, for machine translation in NLP or for evaluating model performance on safety-critical motion planning tasks in robotics. As future work, we propose adapting BAYES-TREX to incorporate temporal output dependencies into the inference formulation.

Ethics Statement. BAYES-TREX has potential to allow humans to build more accurate mental models of how neural networks make decisions. Further, BAYES-TREX can be useful for debugging, interpreting, and understanding networks—all of which can help us build *better*, less biased, increasingly human-aligned models. However, BAYES-TREX is subject to the same caveats as typical software testing approaches: the absence of exposed bad samples does not mean the system is free from defects. One concern is how system designers and users will interact with BAYES-TREX in practice. If BAYES-TREX does not reveal degenerate examples, these stakeholders might develop inordinate trust (Lee and See 2004) in their models.

Additionally, one BAYES-TREX use case is to generate examples for use with downstream local explanation methods. As a community, we know many of these methods can be challenging to understand (Olah, Mordvintsev, and Schubert 2017; Nguyen, Yosinski, and Clune 2019), misleading (Adebayo et al. 2018; Kindermans et al. 2019; Rudin 2019), or susceptible to adversarial attacks (Slack et al. 2019). In human-human interaction, even nonsensical explanations can increase compliance (Langer, Blank, and Chanowitz 1978). As we build post-hoc explanation techniques, we must evaluate whether the produced explanations help humans moderate trust and act appropriately—for example, by overriding the model's decisions.

References

Adebayo, J.; Gilmer, J.; Muelly, M.; Goodfellow, I.; Hardt, M.; and Kim, B. 2018. Sanity checks for saliency maps. In *NeurIPS*, 9505–9515.

Arjovsky, M.; Bottou, L.; Gulrajani, I.; and Lopez-Paz, D. 2019. Invariant risk minimization. *arXiv:1907.02893* .

Arora, S.; and Zhang, Y. 2017. Do GANs actually learn the distribution? An empirical study. *arXiv:1706.08224* .

Bau, D.; Zhou, B.; Khosla, A.; Oliva, A.; and Torralba, A. 2017. Network dissection: Quantifying interpretability of deep visual representations. In *CVPR*, 6541–6549.

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. *JMLR* .

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural networks. In *ICML*, 1613–1622.

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.

Chen, C.; Li, O.; Tao, D.; Barnett, A.; Rudin, C.; and Su, J. K. 2019. This looks like that: deep learning for interpretable image recognition. In *NeurIPS*, 8928–8939.

Cusumano-Towner, M.; and Mansinghka, V. K. 2017. AIDE: An algorithm for measuring the accuracy of probabilistic inference algorithms. In *NeurIPS*.

DeVries, T.; and Taylor, G. W. 2018. Learning confidence for out-of-distribution detection in neural networks. *arXiv:1802.04865* .

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

Erhan, D.; Bengio, Y.; Courville, A.; and Vincent, P. 2009. Visualizing higher-layer features of a deep network. *University of Montreal* 1341(3): 1.

Fremont, D. J.; Dreossi, T.; Ghosh, S.; Yue, X.; Sangiovanni-Vincentelli, A. L.; and Seshia, S. A. 2019. Scenic: a language for scenario specification and scene generation. In *PLDI*.

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

Ghorbani, A.; Wexler, J.; and Kim, B. 2019. Automating interpretability: Discovering and testing visual concepts learned by neural networks. *arXiv:1902.03129* .

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

Gorham, J.; and Mackey, L. 2015. Measuring sample quality with Stein's method. In *NeurIPS*, 226–234.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On calibration of modern neural networks. In *ICML*.

Hendrycks, D.; and Gimpel, K. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv:1610.02136* .

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 6626–6637.

Hoffman, M. D.; and Gelman, A. 2014. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *JMLR* 15(1): 1593–1623.

Johnson, J.; Hariharan, B.; van der Maaten, L.; Fei-Fei, L.; Zitnick, C. L.; and Girshick, R. 2017. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. In *CVPR*.

Kim, J.; Ricci, M.; and Serre, T. 2018. Not-So-CLEVR: learning same–different relations strains feedforward neural networks. *Interface focus* 8(4): 20180011.

Kindermans, P.-J.; Hooker, S.; Adebayo, J.; Alber, M.; Schütt, K. T.; Dähne, S.; Erhan, D.; and Kim, B. 2019. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, 267–280. Springer.

Koh, P. W.; and Liang, P. 2017. Understanding black-box predictions via influence functions. In *ICML*.

Langer, E. J.; Blank, A.; and Chanowitz, B. 1978. The mindlessness of ostensibly thoughtful action: The role of "placebic" information in interpersonal interaction. *Journal of personality and social psychology* 36(6): 635.

LeCun, Y.; and Cortes, C. 2010. MNIST handwritten digit database URL http://yann.lecun.com/exdb/mnist/.

Lee, J. D.; and See, K. A. 2004. Trust in automation: Designing for appropriate reliance. *Human factors* 46(1): 50–80.

Lee, K.; Lee, H.; Lee, K.; and Shin, J. 2017. Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples.

Li, O.; Liu, H.; Chen, C.; and Rudin, C. 2018. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Liang, S.; Li, Y.; and Srikant, R. 2017. Principled detection of out-of-distribution examples in neural networks. *arXiv:1706.02690* .

Lipton, Z. C. 2018. The mythos of model interpretability. *Queue* 16(3): 31–57.

Long, M.; Cao, Y.; Wang, J.; and Jordan, M. I. 2015. Learning transferable features with deep adaptation networks. *arXiv:1502.02791* .

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. In *NeurIPS*, 4765–4774.

Maaten, L. v. d.; and Hinton, G. 2008. Visualizing data using t-SNE. *JMLR* 9(Nov): 2579–2605.

Mitchell, M.; Wu, S.; Zaldivar, A.; Barnes, P.; Vasserman, L.; Hutchinson, B.; Spitzer, E.; Raji, I. D.; and Gebru, T. 2019. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, 220–229.

Neal, R. M.; et al. 2011. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo* 2(11): 2.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning .

Nguyen, A.; Dosovitskiy, A.; Yosinski, J.; Brox, T.; and Clune, J. 2016. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *NeurIPS*, 3387–3395.

Nguyen, A.; Yosinski, J.; and Clune, J. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, 427–436.

Nguyen, A.; Yosinski, J.; and Clune, J. 2019. Understanding neural networks via feature visualization: A survey. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer.

Odena, A.; Olsson, C.; Andersen, D.; and Goodfellow, I. 2019. TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing. In *ICML*, 4901–4911. Long Beach, California, USA.

Olah, C.; Mordvintsev, A.; and Schubert, L. 2017. Feature Visualization. *Distill* doi:10.23915/distill.00007. Https://distill.pub/2017/feature-visualization.

Peng, X. B.; Andrychowicz, M.; Zaremba, W.; and Abbeel, P. 2018. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, 1–8. IEEE.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why should I trust you?" Explaining the predictions of any classifier. In *KDD*, 1135–1144.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1(5): 206–215.

Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2013. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *CoRR* abs/1312.6034.

Slack, D.; Hilgard, S.; Jia, E.; Singh, S.; and Lakkaraju, H. 2019. How can we fool LIME and SHAP? Adversarial Attacks on Post hoc Explanation Methods. *arXiv:1911.02508* .

Smilkov, D.; Thorat, N.; Kim, B.; Viégas, F.; and Wattenberg, M. 2017. Smoothgrad: removing noise by adding noise. *arXiv:1706.03825* .

Song, Y.; and Ermon, S. 2019. Generative Modeling by Estimating Gradients of the Data Distribution.

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

Tan, J.; Zhang, T.; Coumans, E.; Iscen, A.; Bai, Y.; Hafner, D.; Bohez, S.; and Vanhoucke, V. 2018. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv:1804.10332* .

Thulasidasan, S.; Chennupati, G.; Bilmes, J. A.; Bhattacharya, T.; and Michalak, S. 2019. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *NeurIPS*, 13888–13899.

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 23–30. IEEE.

Tzeng, E.; Hoffman, J.; Saenko, K.; and Darrell, T. 2017. Adversarial discriminative domain adaptation. In *CVPR*.

Tzeng, E.; Hoffman, J.; Zhang, N.; Saenko, K.; and Darrell, T. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv:1412.3474* .

Vernekar, S.; Gaurav, A.; Abdelzad, V.; Denouden, T.; Salay, R.; and Czarnecki, K. 2019. Out-of-distribution Detection in Classifiers via Generation. *arXiv:1910.04241* .

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747* .

Zeiler, M. D.; and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, 818–833. Springer.

Zhao, Z.; Dua, D.; and Singh, S. 2018. Generating Natural Adversarial Examples. In *ICLR*.

BAYES-TREX: Model Transparency by Example – Appendix

A More Details on the Test Set Evaluations

We compare the insights we can derive with BAYES-TREX to using the test data for evaluation. In particular, we compare high confidence classification failures and ambiguous examples, as these targets have natural test set analogues. We find missing or sparse representation of these cases in the evaluated test sets for MNIST, Fashion-MNIST, and CLEVR. For high confidence classification failures, we find this test set analysis exposes more mislabeled examples than true misclassifications. These mislabeled examples are not useful for our task of holistic model analysis.

High Confidence Classification Failures

Classification failures with undue high confidence are especially harmful. In human-AI teaming, such high confidence may result in the human not critically reconsidering the model's prediction. For sensitive domains like AI-assisted healthcare, these model failures can have catastrophic consequences.

One way to evaluate classification failures is to compare BAYES-TREX's findings to insights we can derive using a standard confusion matrix evaluation. Note that here the confusion matrix consists of all test set examples, not just high confidence examples. In particular, we expect these confusion matrices to expose common class confusions. For example, we might expect a well-performing classifier to confuse semantically similar classes such as shirts and T-shirts or sneakers and ankle boots in the Fashion-MNIST dataset. One insight we discovered when applying BAYES-TREX to the Fashion-MNIST dataset is that chunky shoes like ankle boots are often misclassified as bags. Looking at the confusion matrix, this particular trend is not apparent: 0 ankle boot examples are misclassified as bags (Figure 9).

We next evaluate the *high confidence* classification failures using the test set. For MNIST, out of 10,000 test images, we find that 84 have high confidence misclassifications. However, on further examination, we find that the majority (60/84) of these images are mislabeled, and are not true misclassifications. See Table 4. For example, all of the ostensibly misclassified 2s clearly belong to other classes (8, 7, 7, 3, 1, 7, 7, 7, respectively). Fashion-MNIST had more high confidence misclassifications—802 in total. We randomly select 10 misclassifications from each class for hand labeling (with the exception of the "trousers" class, as there were only 3 total misclassifications for this label). Again, we find the majority of these ostensibly misclassified examples to be caused by mislabeled ground truth (52/93), though these results are more balanced than MNIST. Finding these mislabeled examples is not useful for holistic classifier evaluation.

Lastly, using the CLEVR test set of $15,000$ images, we find no examples which meet the 2 cylinder failure constraint, while BAYES-TREX is comparatively successful in this task. We further find only sparse examples for the other 3 targets, between 5 and 15 examples for each. See Table 3.

Ambiguous Examples

To find ambiguous examples using the test data, we search for examples with classifier confidence in the range $40\% - 60\%$ for two classes. Of $10,000$ MNIST test images, 12 test set examples received ambiguous classifications. Two of these examples were from the 9v4 class and two were from the 9v7 class. While FashionMNIST had 162 ambiguous images, these images were again distributed across just 12 class combinations. In comparison, BAYES-TREX found ambiguous examples for 29 additional MNIST class combinations and 18 additional Fashion-MNIST class combinations. Figure 10 shows a side-by-side comparison of these ambiguous test set examples alongside the BAYES-TREX-uncovered ambiguous examples.

Table 3: Number of true misclassified test set examples which meet the specified constraints. While CLEVR and MNIST results are raw totals from the test set evaluations, Fashion-MNIST is sampled out of 93 total misclassifications.

	CLEVR Misclassifications				MNIST Misclassifications										Fashion-MNIST Misclassifications									
	1 Sphere	1 Cube	1 Cylinder	2 Cylinders	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
# Examples	5	8	15	0	3	3	0	5	3	1	3	4	0	2	2	0	9	4	9	1	3	2	1	10

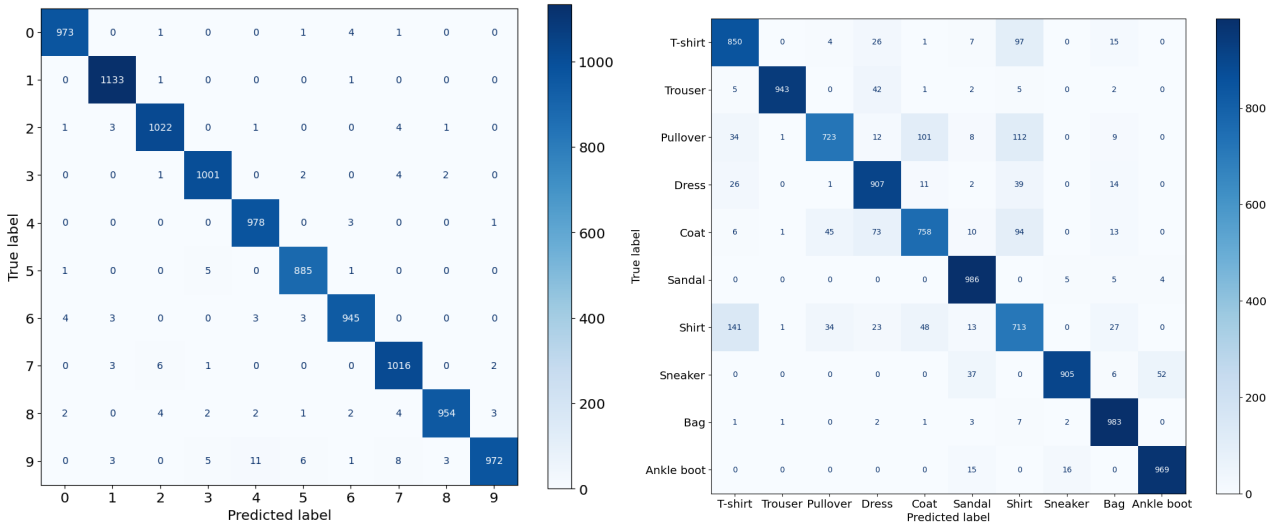
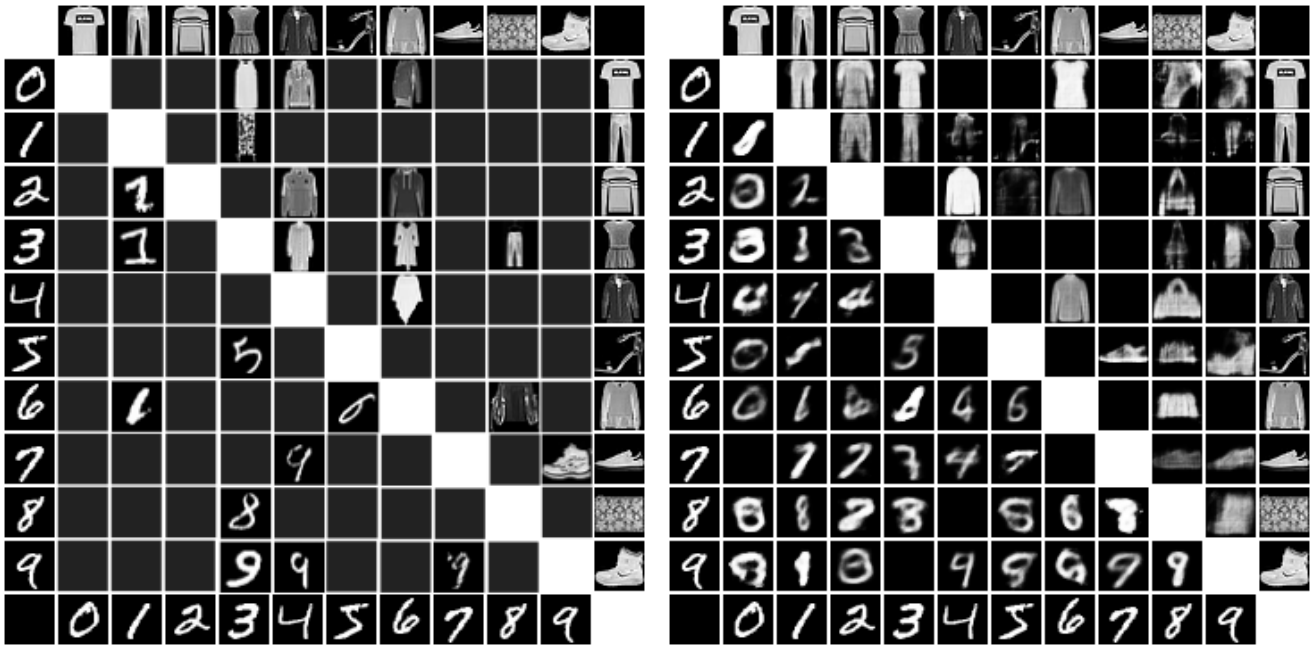


Figure 9: Confusion Matrices for MNIST (above) and Fashion-MNIST (below) classifiers. Note that these matrices include all test set examples, not just those which evoke high confidence responses from the classifier.



(a) Test Set (b) BAYES-TREX

Figure 10: 10(a) shows ambiguous examples present in the MNIST and FashionMNIST test sets. 10(b) shows ambiguous examples generated with BAYES-TREX. We note that BAYES-TREX has higher coverage of class pairings on both MNIST and FashionMNIST. Black squares indicate that no such examples were found.

Table 4: An alternative to using BAYES-TREX for finding highly confident classification failures is to evaluate the high confidence example confusion matrix and associated images from the test set. Here, we show all 'misclassified' examples where the classifier failed to predict the given label for the MNIST and Fashion-MNIST datasets. For MNIST, we observe that the majority (60/84) of these images are mislabeled: for example, all of the labeled 2s clearly belong to other classes (8, 7, 7, 3, 1, 7, 7, 7, respectively). While MNIST had 84 total misclassifications, Fashion-MNIST had 802 total misclassifications. We randomly select 10 misclassifications from each class for analysis (with the exception of the "trousers" class, as there were 3 total misclassifications for this label). While Fashion-MNIST is more balanced, we again observe a majority of examples to be mislabeled ground truth (52/93) instead of misclassifications.

Class	Misclassified	Mislabeled
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
Tshirt		
Trouser		
Pullover		
Dress		
Coat		
Sandal		
Shirt		
Sneaker		
Bag		
Boot		

B Network Architecture for MNIST & Fashion-MNIST

For all experiments on MNIST and Fashion-MNIST, the VAE architecture is shown in Table 5 (left), and the GAN architecture is shown in Table 5 (right). For all experiments on MNIST and Fashion-MNIST except for the domain adaptation analysis, the classifier architecture is shown in Table 6 (left). The classifier used in the domain adaptation analysis is the LeNet architecture, following the provided source code, shown in Table 6 (right). All experiments are performed with an NVIDIA GeForce 1080 GPU. We use binary cross entropy loss for training the generative models, and negative log likelihood loss for training the classifiers. We additionally submit our open source code as supplemental.

Table 5: Left: VAE architecture; right: GAN architecture.

Encoder input: $28 \times 28 \times 1$
Flatten
Fully-connected 784×400
ReLU
Mean: Fully-connected 400×5
Log-variance: Fully-connected 400×5
Decoder input: 5 (latent dimension)
Fully-connected 5×400
ReLU
Fully-connected 400×784
Reshape $28 \times 28 \times 1$
Sigmoid

Input: 5 (latent dimension)
Reshape $1 \times 1 \times 5$
Conv-transpose: 512 filters, size=4×4, stride = 1
Batch-norm, ReLU
Conv-transpose: 256 filters, size=4×4, stride = 2
Batch-norm, ReLU
Conv-transpose: 128 filters, size=4×4, stride = 2
Batch-norm, ReLU
Conv-transpose: 64 filters, size=4×4, stride = 2
Batch-norm, ReLU
Conv-transpose: 1 filters, size=1×1, stride = 1
Sigmoid

Table 6: Left: classifier architecture in all experiments except domain adaptation analysis; right: LeNet classifier architecture in domain adaptation analysis (used in code released by ADDA authors).

Input: $28 \times 28 \times 1$
Conv: 32 filters, size = 3×3, stride = 1
ReLU
Conv: 64 filters, size = 3×3, stride = 1
Drop-out, prob = 0.25
Max-pool, size = 2×2
Flatten
Fully-connected 9216×128
ReLU
Drop-out, prob = 0.5
Fully-connected 128×10
Soft-max

Input: $28 \times 28 \times 1$
Conv: 20 filters, size = 5×5, stride = 1
ReLU
Max-pool, size = 2×2
Conv: 50 filters, size = 5×5, stride = 1
ReLU
Max-pool, size = 2×2
Flatten
Fully-connected 800×500
ReLU
Fully-connected 500×10
Soft-max

C Evaluating Learned Data Distribution Fit: Fréchet Inception Distance (FID) Scores

Fréchet Inception Distance (FID) scores are one measure to evaluate how well a learned data distribution fits the ground truth distribution. We report FID scores for all learned generative models in Table 7. We note that GANs consistently outperform VAEs for all data distributions. We further note that all generative models perform better on MNIST data than on Fashion-MNIST. MNIST GANs are the best performing models; Fashion-MNIST VAEs are the worst performing models. We measure FID for each VAE and GAN. In particular, these include the generative models trained for exposing in-distribution adversarial examples by leaving out one class at a time. Further, these FID scores include generative models trained on select subsets of the data for novel class extrapolation studies. For example, one GAN is trained only on the MNIST digits $\{2, 4, 5, 7, 8\}$.

Table 7: Fréchet Inception Distance (FID) scores for all learned data distributions; a lower score indicates a better distribution fit. We note that GAN performance is markedly superior to VAE performance, and MNIST performance is superior to Fashion-MNIST performance across all models. Results are computed across 1000 samples. Fashion-MNIST class mapping: [0: T-shirt, 1: Trouser, 2: Pullover, 3: Dress, 4: Coat, 5: Sandal, 6: Shirt, 7: Sneaker, 8: Bag, 9: Ankle boot].

Model	Dataset	Subset	FID
GAN	MNIST	All	11.83
		Without 0	12.10
		Without 1	12.08
		Without 2	13.57
		Without 3	12.71
		Without 4	12.25
		Without 5	12.21
		Without 6	11.86
		Without 7	11.64
		Without 8	12.31
		Without 9	12.34
		$\{2, 4, 5, 7, 8\}$	13.45
	Fashion	All	29.44
		Without 0	28.91
		Without 1	31.18
		Without 2	30.11
		Without 3	28.95
		Without 4	30.43
		Without 5	27.67
		Without 6	29.68
		Without 7	28.56
		Without 8	30.87
		Without 9	29.22
		$\{0, 1, 4, 7, 8\}$	33.11

Model	Dataset	Subset	FID
VAE	MNIST	All	72.33
		Without 0	71.28
		Without 1	75.36
		Without 2	64.77
		Without 3	63.66
		Without 4	66.96
		Without 5	63.31
		Without 6	67.64
		Without 7	62.45
		Without 8	64.14
		Without 9	66.57
		—	—
	Fashion	All	87.89
		Without 0	89.21
		Without 1	92.02
		Without 2	91.20
		Without 3	85.51
		Without 4	88.38
		Without 5	84.17
		Without 6	85.58
		Without 7	84.93
		Without 8	83.66
		Without 9	81.48
		—	—

D Full Quantitative Confidence Results

Tables 8, 9, and 10 present the full extension of Table 1 in the main text. These results show that the inferred samples have predicted confidence closely matching the specified confidence targets. This indicates the MCMC methods used by BAYES-TREX are successful for the tested domains and scenarios. Queries for 5 Cubes in the novel class extrapolation CLEVR experiments use a stopping criterion of 1500 samples instead of the standard 500. Averages reported across 10 inference runs.

Table 8: Full table of measured prediction confidence values for BAYES-TREX samples on high-confidence examples (left) and high confidence misclassifications (right). Accompanying visual results in Appendices E and I.

Target	Prediction Confidence
$P_0 = 1$	0.999 ± 0.006
$P_1 = 1$	0.999 ± 0.003
$P_2 = 1$	0.999 ± 0.006
$P_3 = 1$	0.999 ± 0.005
$P_4 = 1$	0.998 ± 0.008
$P_5 = 1$	0.999 ± 0.006
$P_6 = 1$	0.998 ± 0.007
$P_7 = 1$	0.998 ± 0.007
$P_8 = 1$	0.999 ± 0.004
$P_9 = 1$	0.998 ± 0.007
$P_{\text{T-Shirt}} = 1$	0.991 ± 0.016
$P_{\text{Trouser}} = 1$	0.999 ± 0.006
$P_{\text{Pullover}} = 1$	0.984 ± 0.019
$P_{\text{Dress}} = 1$	0.993 ± 0.008
$P_{\text{Coat}} = 1$	0.983 ± 0.021
$P_{\text{Sandal}} = 1$	0.998 ± 0.008
$P_{\text{Shirt}} = 1$	0.987 ± 0.020
$P_{\text{Sneaker}} = 1$	0.994 ± 0.016
$P_{\text{Bag}} = 1$	0.999 ± 0.006
$P_{\text{Ankle Boot}} = 1$	0.996 ± 0.012
$P_{\text{5 Spheres}} = 1$	0.943 ± 0.020
$P_{\text{2 Blue Spheres}} = 1$	0.892 ± 0.245

Target	Prediction Confidence
$P_0 = 1$	0.981 ± 0.027
$P_1 = 1$	0.953 ± 0.028
$P_2 = 1$	0.968 ± 0.028
$P_3 = 1$	0.969 ± 0.027
$P_4 = 1$	0.955 ± 0.030
$P_5 = 1$	0.990 ± 0.018
$P_6 = 1$	0.970 ± 0.026
$P_7 = 1$	0.968 ± 0.029
$P_8 = 1$	0.982 ± 0.024
$P_9 = 1$	0.983 ± 0.022
$P_{\text{T-Shirt}} = 1$	0.964 ± 0.029
$P_{\text{Trouser}} = 1$	(sample failure)
$P_{\text{Pullover}} = 1$	0.886 ± 0.027
$P_{\text{Dress}} = 1$	0.970 ± 0.026
$P_{\text{Coat}} = 1$	0.938 ± 0.030
$P_{\text{Sandal}} = 1$	0.968 ± 0.030
$P_{\text{Shirt}} = 1$	0.938 ± 0.032
$P_{\text{Sneaker}} = 1$	0.969 ± 0.028
$P_{\text{Bag}} = 1$	0.967 ± 0.026
$P_{\text{Ankle Boot}} = 1$	0.971 ± 0.027
$P_{\text{1 Cube}} = 1$	0.929 ± 0.062
$P_{\text{1 Cylinder}} = 1$	0.972 ± 0.021
$P_{\text{1 Sphere}} = 1$	0.843 ± 0.266
$P_{\text{2 Cylinders}} = 1$	0.545 ± 0.230

Table 9: Full table for graded confidence examples for MNIST (left) and Fashion-MNIST (right); accompanying visual results are presented in Appendix G.

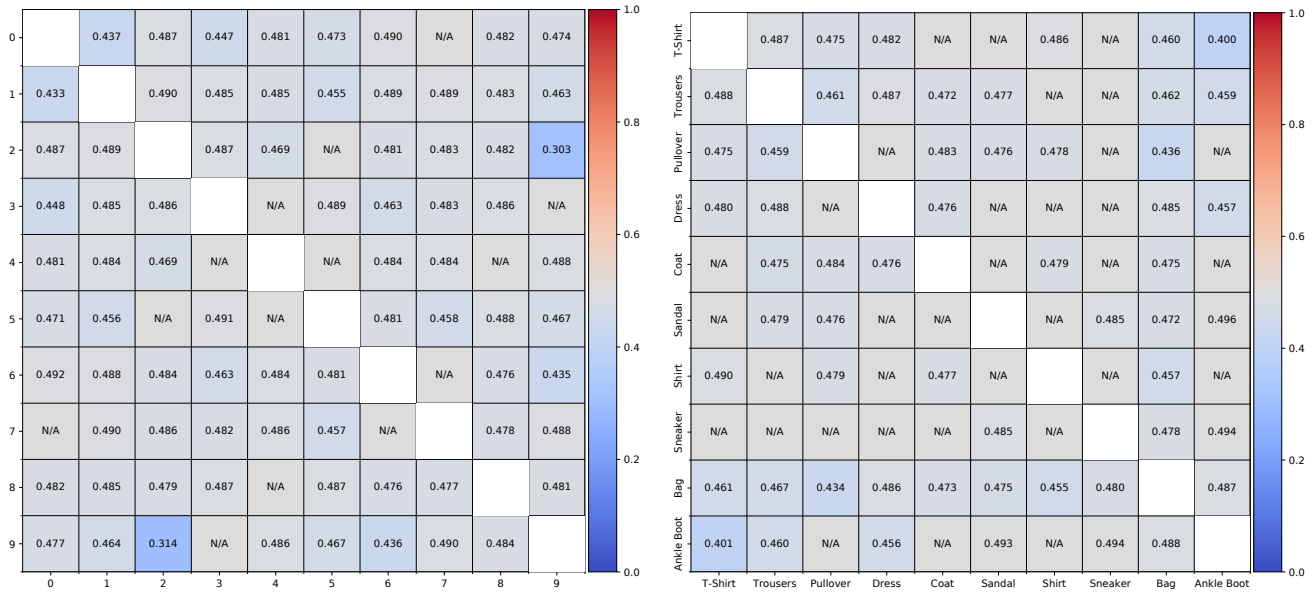
Target	Prediction Confidence
$P_8 = 0.0, P_9 = 1.0$	$(0.002 \pm 0.006, 0.990 \pm 0.016)$
$P_8 = 0.1, P_9 = 0.9$	$(0.030 \pm 0.039, 0.936 \pm 0.051)$
$P_8 = 0.2, P_9 = 0.8$	$(0.170 \pm 0.039, 0.788 \pm 0.040)$
$P_8 = 0.3, P_9 = 0.7$	$(0.275 \pm 0.041, 0.682 \pm 0.040)$
$P_8 = 0.4, P_9 = 0.6$	$(0.378 \pm 0.040, 0.578 \pm 0.040)$
$P_8 = 0.5, P_9 = 0.5$	$(0.477 \pm 0.039, 0.477 \pm 0.039)$
$P_8 = 0.6, P_9 = 0.4$	$(0.581 \pm 0.038, 0.374 \pm 0.039)$
$P_8 = 0.7, P_9 = 0.3$	$(0.680 \pm 0.041, 0.275 \pm 0.039)$
$P_8 = 0.8, P_9 = 0.2$	$(0.788 \pm 0.040, 0.167 \pm 0.041)$
$P_8 = 0.9, P_9 = 0.1$	$(0.926 \pm 0.050, 0.039 \pm 0.040)$
$P_8 = 1.0, P_9 = 0.0$	$(0.989 \pm 0.016, 0.002 \pm 0.007)$

Target	Prediction Confidence
$P_{\text{T-Shirt}} = 0.0, P_{\text{Trousers}} = 1.0$	$(0.001 \pm 0.004, 0.995 \pm 0.012)$
$P_{\text{T-Shirt}} = 0.1, P_{\text{Trousers}} = 0.9$	$(0.026 \pm 0.035, 0.950 \pm 0.050)$
$P_{\text{T-Shirt}} = 0.2, P_{\text{Trousers}} = 0.8$	$(0.166 \pm 0.040, 0.791 \pm 0.041)$
$P_{\text{T-Shirt}} = 0.3, P_{\text{Trousers}} = 0.7$	$(0.275 \pm 0.037, 0.686 \pm 0.038)$
$P_{\text{T-Shirt}} = 0.4, P_{\text{Trousers}} = 0.6$	$(0.379 \pm 0.038, 0.586 \pm 0.038)$
$P_{\text{T-Shirt}} = 0.5, P_{\text{Trousers}} = 0.5$	$(0.436 \pm 0.040, 0.459 \pm 0.040)$
$P_{\text{T-Shirt}} = 0.6, P_{\text{Trousers}} = 0.4$	$(0.583 \pm 0.038, 0.382 \pm 0.037)$
$P_{\text{T-Shirt}} = 0.7, P_{\text{Trousers}} = 0.3$	$(0.685 \pm 0.039, 0.281 \pm 0.040)$
$P_{\text{T-Shirt}} = 0.8, P_{\text{Trousers}} = 0.2$	$(0.790 \pm 0.037, 0.177 \pm 0.037)$
$P_{\text{T-Shirt}} = 0.9, P_{\text{Trousers}} = 0.1$	$(0.936 \pm 0.045, 0.029 \pm 0.041)$
$P_{\text{T-Shirt}} = 1.0, P_{\text{Trousers}} = 0.0$	$(0.985 \pm 0.019, 0.000 \pm 0.003)$

Table 10: Full table for out-of-distribution, novel class extrapolation (left) and domain adaptation (right) samples (using the ADDA model). Accompanying visual results in Appendices J and K.

Target	Prediction Confidence
$P_0 = 1$	0.976 ± 0.025
$P_1 = 1$	0.988 ± 0.186
$P_3 = 1$	0.987 ± 0.020
$P_6 = 1$	0.989 ± 0.018
$P_9 = 1$	0.995 ± 0.013
$P_{\text{Pullover}} = 1$	0.991 ± 0.016
$P_{\text{Dress}} = 1$	0.994 ± 0.013
$P_{\text{Sandal}} = 1$	0.995 ± 0.013
$P_{\text{Shirt}} = 1$	0.994 ± 0.012
$P_{\text{Ankle Boot}} = 1$	0.993 ± 0.015
$P_{\text{1 Cube}} = 1$	0.983 ± 0.014
$P_{\text{1 Cylinder}} = 1$	0.959 ± 0.031
$P_{\text{1 Sphere}} = 1$	0.969 ± 0.022
$P_{\text{5 Cubes}} = 1$	0.921 ± 0.029

Target	Prediction Confidence
$P_0 = 1$	0.996 ± 0.011
$P_1 = 1$	0.994 ± 0.014
$P_2 = 1$	0.998 ± 0.008
$P_3 = 1$	0.994 ± 0.015
$P_4 = 1$	0.997 ± 0.010
$P_5 = 1$	0.998 ± 0.007
$P_6 = 1$	0.996 ± 0.011
$P_7 = 1$	0.996 ± 0.011
$P_8 = 1$	0.995 ± 0.013
$P_9 = 1$	0.996 ± 0.012

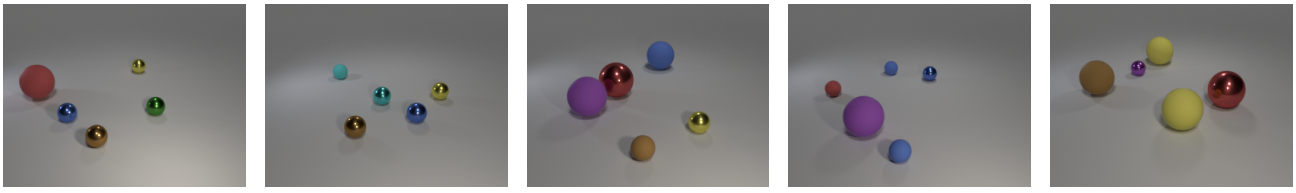


(a) MNIST (b) Fashion-MNIST

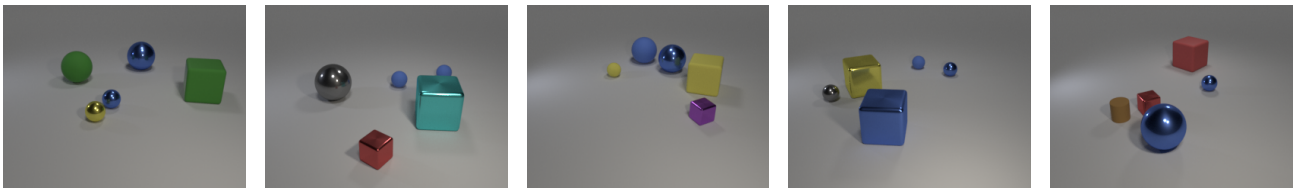
Figure 11: Full table for ambiguous examples, for both the MNIST and Fashion-MNIST datasets. For each class combination, the lower triangle shows the the confidence for the digit denoted on the x-axis, and the upper triangle shows the confidence for the digit on the y-axis. For example, for the MNIST class combination 9v0, the classifier confidence that the digit is a 0 is 0.477 (very bottom left, 11(a)) while the classifier confidence that the digit is a 9 is 0.474 (very top right, 11(a)).

E More Details on the High-Confidence Example Experiment

The first check we perform with BAYES-TREX is to infer high confidence examples while using the full training and test distribution for generation. Figure 12 presents additional high-confidence examples for CLEVR. Figure 13 presents additional high-confidence examples for MNIST and Fashion-MNIST.



(a) $P_{\text{5 Sph.}} = 94.8\%$ (b) $P_{\text{5 Sph.}} = 94.5\%$ (c) $P_{\text{5 Sph.}} = 94.6\%$ (d) $P_{\text{5 Sph.}} = 95.2\%$ (e) $P_{\text{5 Sph.}} = 92.0\%$



(f) $P_{\text{2 Blue}} = 96.3\%$ (g) $P_{\text{2 Blue}} = 96.1\%$ (h) $P_{\text{2 Blue}} = 94.9\%$ (i) $P_{\text{2 Blue}} = 96.8\%$ (j) $P_{\text{2 Blue}} = 97.8\%$

Figure 12: Above, 12(a)–12(e): selected examples classified as containing 5 spheres with high confidence. Below, 12(f)–12(j): selected examples classified as containing 2 blue spheres with high confidence.

(a) MNIST (b) Fashion-MNIST

Figure 13: High-confidence examples from MNIST and Fashion-MNIST. There are no misclassifications. MNIST columns represent digit 0 to 9, respectively. Fashion-MNIST columns represent t-shirt, trousers, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot, respectively.

F More Details on the Ambiguous Example Experiments

Another BAYES-TREX use case is to qualitatively evaluate class boundaries by finding ambiguous examples. Figure 5 in the main text presents a matrix showing one sampled image for each pair of classes from both MNIST and Fashion-MNIST (e.g., Digit 1 vs. Digit 7; T-shirt vs. Pullover). Figure 14 presents additional breakdowns for two pairs, Digit 1 vs. Digit 7 from MNIST and T-shirt vs. Pullover from Fashion-MNIST. The violin plot confirms that the neural network is indeed making the ambiguous target predictions, and the latent visualization indicates that the samples lie around the class boundaries and are in-distribution (i.e., having close proximity to others from the prior in the latent visualization with T-SNE (Maaten and Hinton 2008) dimensionality reduction).

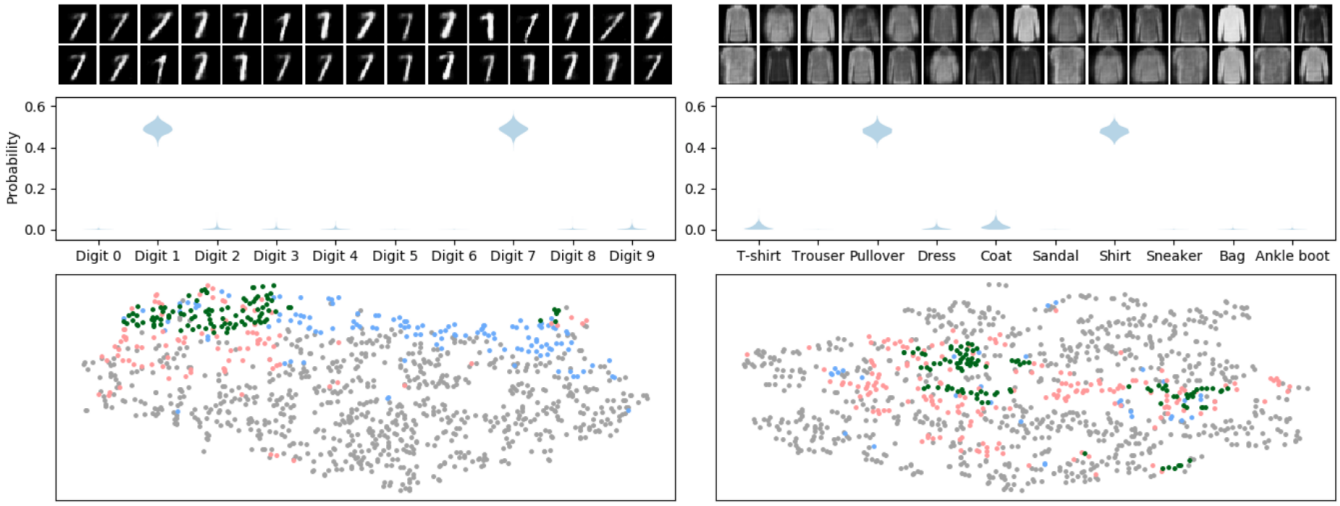


Figure 14: Left: ambiguous samples for digit 1 vs. 7 in MNIST. Right: ambiguous samples for pullover vs. shirt in Fashion-MNIST. Top: 30 sampled images. Middle: the ambiguous predictions made by the classifier. Bottom: latent space visualization. Green dots represent ambiguous samples from the posterior, red and blue dots represents samples from the prior that are predicted by the classifier to be either class of interest, and gray dots represents other samples from the prior. The ambiguous samples are on the class boundaries.

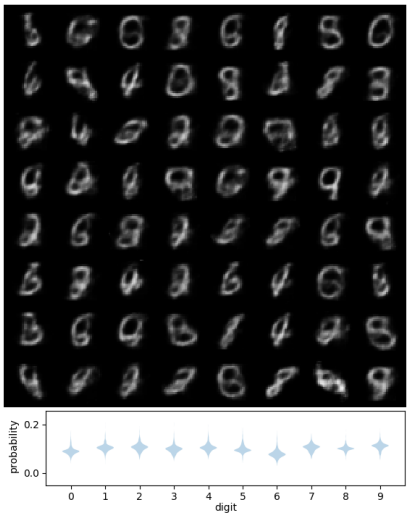


Figure 15: In addition to binary ambiguity, BAYES-TREX can be used to find examples which are ambiguous between more than two classes. Here we show uniformly ambiguous examples for MNIST and their target predictions. While the resultant images seem especially blurry, note that this task is somewhat contrived: a human would not necessarily know how to draw a uniformly ambiguous digit.

G More Details on the Graded-Confidence Example Experiments

BAYES-TREX can be used to sample confidence predictions which interpolate between classes. In Figure 16, we show MNIST samples which interpolate from $(P_8 = 1.0, P_9 = 0.0)$ to $(P_8 = 0.0, P_9 = 1.0)$ and Fashion-MNIST samples from $(P_{\text{T-shirt}} = 1.0, P_{\text{Trousers}} = 0.0)$ to $(P_{\text{T-shirt}} = 0.0, P_{\text{Trousers}} = 1.0)$ over intervals of 0.1, using a VAE as the generator. The target probability for other classes is 0.

By interpolating between two quite different classes, we can gain some insight into the model's behaviour. For example, in Figure 16, we see that the interpolation from 8 to 9 generally shrinks the bottom circle toward a stroke, which is the key difference between digits 8 and 9 to be the width of the bottom circle. For Fashion-MNIST, we consider the case of T-shirt vs. Trousers. We uncover that the presence of two legs is important for trousers classification, even appearing in samples with $(\mathbf{p}_{\text{T-shirt}} = 0.9, \mathbf{p}_{\text{Trousers}} = 0.1)$ (second column); by contrast, a wider top and the appearance of sleeves are important properties for T-shirt classification: most of the interpolated samples have a short sleeve on the top but two distinct legs on the bottom.

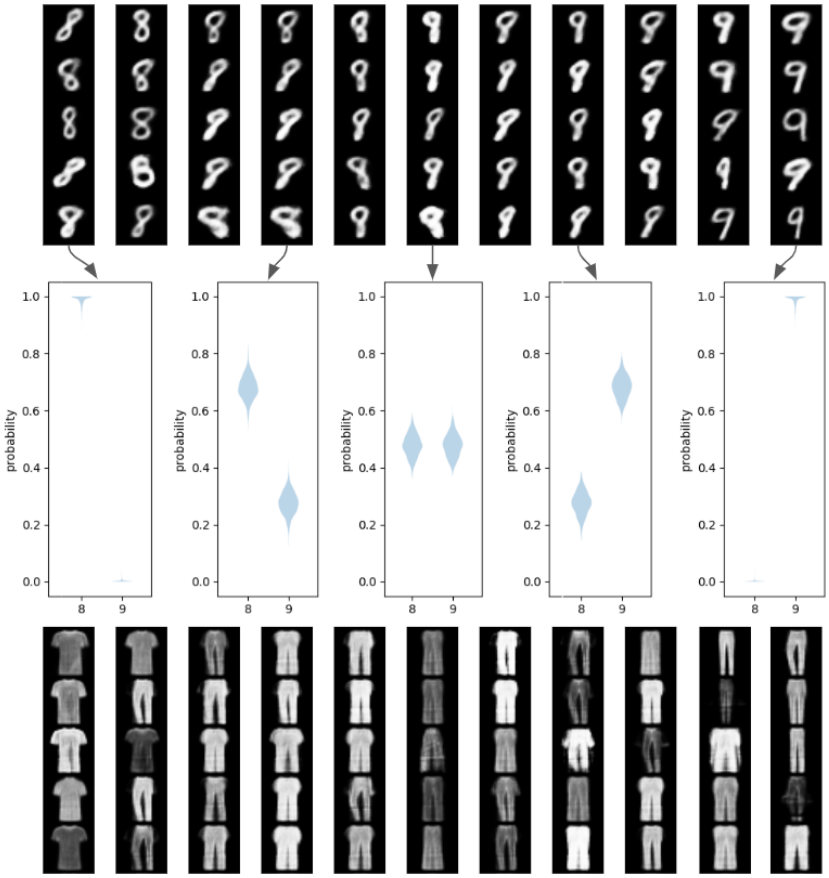


Figure 16: Confidence interpolation between digit 8 and 9 for MNIST and between T-shirt and trousers for Fashion-MNIST. Each of the 11 columns show samples of confidence ranging from $[P_{\text{class a}} = 1.0, P_{\text{class b}} = 0.0]$ (left) to $[P_{\text{class a}} = 0.0, P_{\text{class b}} = 1.0]$ (right), with an interval of 0.1. Select confidence plots for MNIST samples are shown in the 2nd row.

H More Details on the Investigation into GAN Sampling Failure

We failed to sample binary ambivalent and graded confidence (Fashion-)MNIST examples using a GAN as the generative distribution. There are two possible explanations:

1. The formulation of the posterior inference problem somehow is not suitable for a GAN-induced distribution, and/or a Hamiltonian Monte-Carlo sampler cannot easily sample from the posterior; or
2. There are no GAN-generated examples that match the ambivalent prediction targets.

Empirically, we found that the neural network confidences on GAN images are very extreme (i.e. probability very close to 1 for one class and very close to 0 for all other classes). In comparison, we found that the somewhat blurry images produced by VAE have less extreme prediction confidences. We further confirmed that GANs are better data generators than VAEs for our domains by computing FID scores (Appendix C). Therefore, we believe the GAN is largely unable to produce examples which result in ambivalent predictions. This suggests the classifier is consistently confident on high-quality examples.

To verify this, we train a classifier on MNIST with intentionally binary ambivalent targets. Specifically, the loss on an image $\mathbf{x}$ with groundtruth digit $y \in \{0, 1, ..., 9\}$ is

$$l(y, f(\mathbf{x})) = \mathbb{KL}(P_y, f(\mathbf{x})), \tag{11}$$

$$P_{y,i} = \begin{cases} 0.5 & i = y \text{ or } i = (y+1) \bmod 10, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

In words, we use a KL-divergence loss between the network predicted probability $f(\mathbf{x})$ and a target that is *defined* to be binary ambivalent between digit i and digit $i + 1$ (wrapping around at $10 = 0$).

After training, we find that the classifier is indeed confused between the two respective classes, and the classification accuracy is very close to 50%, as expected. Then we use a GAN to sample image that are ambivalent between digit i and $i + 1$ (mod 10). Indeed, the sampler successfully returns images of class i from the posterior (Fig. 17). The violin plots also confirm that the examples are correctly at the target level set. Note that sampling examples that are binary ambivalent between other pairs of digits i and j still fails because the network simply does not make such predictions on any images drawn from the GAN distribution. Similarly, for this experiment, we cannot sample graded confidence examples with any confidence target other than (0.5, 0.5).

This raises the question of whether the typical MNIST and Fashion-MNIST are overconfident. By definition, a high-performing network is not necessarily overconfident, even when highly confident on all examples (Guo et al. 2017). We conclude that these MNIST and Fashion-MNIST classifiers, with their average accuracies in the ≈99%-range, are appropriately highly confident.

We similarly found BAYES-TREX is unable to sample ambivalent CLEVR examples. The data distribution for CLEVR is rendered from a uniform prior, not learned; as a consequence, the resulting examples are high quality. Unlike MNIST and Fashion-MNIST, the CLEVR network only achieves accuracies of ≈60%. The inability to sample ambiguous examples for the CLEVR network likely indicates overconfidence. Indeed, this problem has been previously observed in CLEVR-style settings (Kim, Ricci, and Serre 2018).

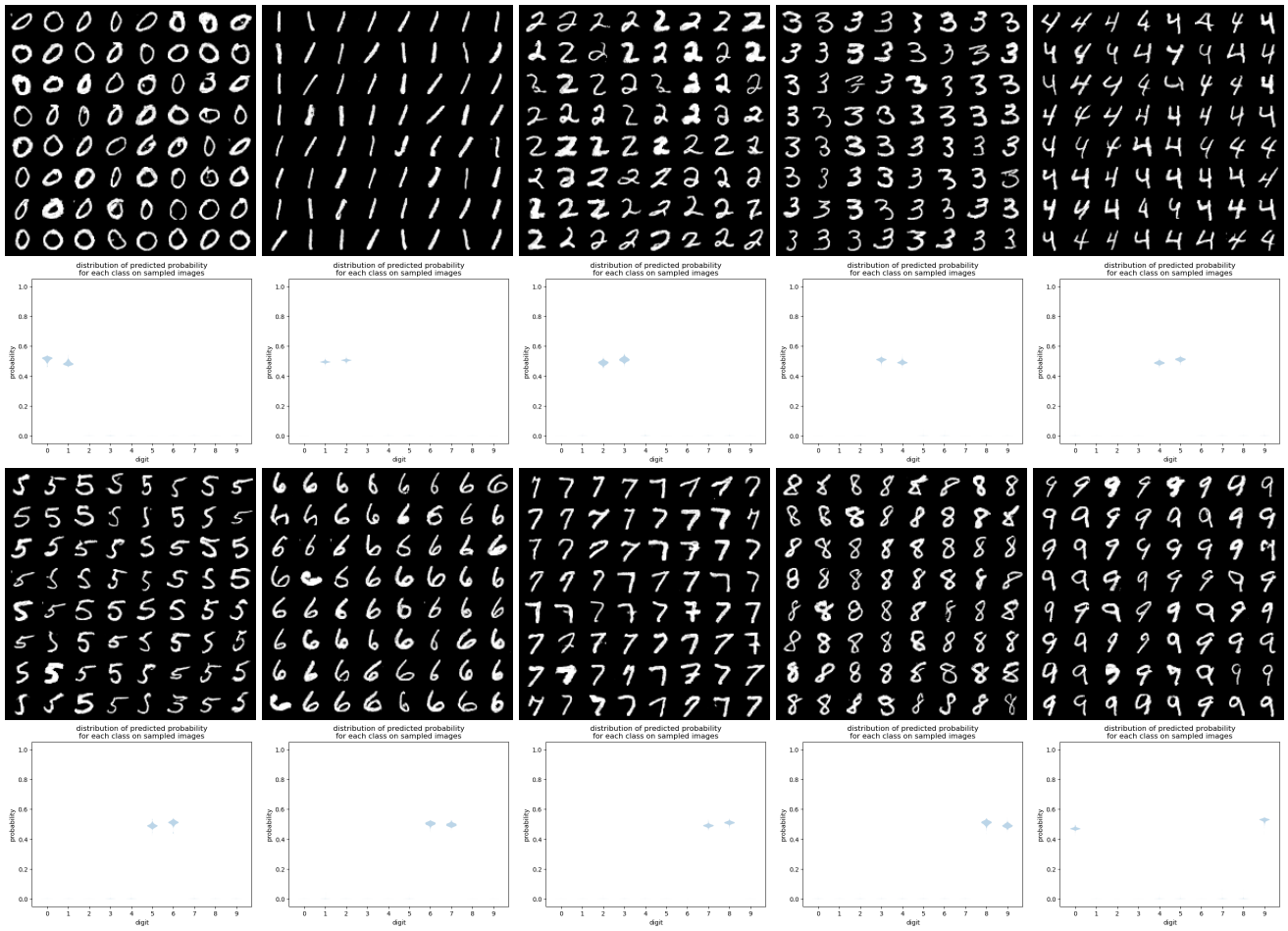


Figure 17: Sampling results with an explicitly ambivalent classifier and a GAN generator. Top 2 rows: digit i vs. $i+1$ for $i \in \{0, 1, 2, 3, 4\}$. Bottom 2 rows: digit i vs. $i+1$ (mod 10) for $i \in \{5, 6, 7, 8, 9\}$.

I More Details on the High-Confidence Misclassification Experiment

In high-stakes scenarios, it is especially important to understand and avoid high-confidence misclassifications. One method for evaluating a model's susceptibility to this is to run the model on the test set and look for such examples. However, since a larger training set usually leads to a higher model performance, the test set size is typically kept relatively small. This smaller test set may be sufficient for providing a low-bias estimate of the generalization performance, but richer analyses are challenging to perform. We show that this test set analysis is insufficient with MNIST and Fashion-MNIST; the discovered high-confidence misclassifications are mostly due to mislabeled test set examples and not due to meaningful misclassifications (Appendix A).

BAYES-TREX provides a means for uncovering such high-confidence misclassification behaviors. Specifically, if we train the classifier on the entire dataset, but we sample highly confident examples with the data distribution excluding a particular class (Figure 3(b)), then the resulting samples are high-confidence misclassifications for that class. Figure 18 shows such examples for CLEVR. For each target inference (e.g. "1 Cube"), we revoked the ability of the generator to produce objects belonging to the target class (e.g., no cubes).

For MNIST and Fashion-MNIST, we sample high-confidence misclassified examples for each class, presented in Fig. 19. Upon inspection, we observe several interesting phenomena. For MNIST, some of the very thin 8s are classified as 1, stretched 9s are classified as 4, and a particular style of 2s and 3s are classified as 7. For Fashion-MNIST, most samples are between semantically similar classes, such as T-shirt $\leftrightarrow$ shirt, and sneaker $\leftrightarrow$ ankle boot. However, we also see that chunky shoes are likely to be classified as bags. This trend of misclassifying chunky shoes as bags is not apparent in any test set evaluations (Appendix A). The sampling procedure failed to find any high-confidence misclassifications for the trousers class.

(a) $P_{\text{1 Cube}} = 96.0\%$ (b) $P_{\text{1 Cube}} = 97.2\%$ (c) $P_{\text{1 Cube}} = 93.5\%$ (d) $P_{\text{1 Cube}} = 67.3\%$ (e) $P_{\text{1 Cube}} = 94.5\%$

(f) $P_{\text{1 Sphere}} = 95.6\%$ (g) $P_{\text{1 Sphere}} = 96.6\%$ (h) $P_{\text{1 Sphere}} = 89.8\%$ (i) $P_{\text{1 Sphere}} = 99.1\%$ (j) $P_{\text{1 Sphere}} = 96.5\%$

(k) $P_{\text{1 Cyl.}} = 90.4\%$ (l) $P_{\text{1 Cyl.}} = 98.6\%$ (m) $P_{\text{1 Cyl.}} = 94.5\%$ (n) $P_{\text{1 Cyl.}} = 96.5\%$ (o) $P_{\text{1 Cyl.}} = 98.5\%$

(p) $P_{\text{2 Cyl.}} = 85.9\%$ (q) $P_{\text{2 Cyl.}} = 60.2\%$ (r) $P_{\text{2 Cyl.}} = 79.4\%$ (s) $P_{\text{2 Cyl.}} = 48.4\%$ (t) $P_{\text{2 Cyl.}} = 60.5\%$

Figure 18: Sampled high confidence misclassified examples and their associated prediction confidences. For each target constraint (e.g., "1 Cube"), the generator is unable to produce examples from the target class (e.g., cubes). The resultant images are composed entirely of non-target-class objects, (e.g., cylinders and spheres).

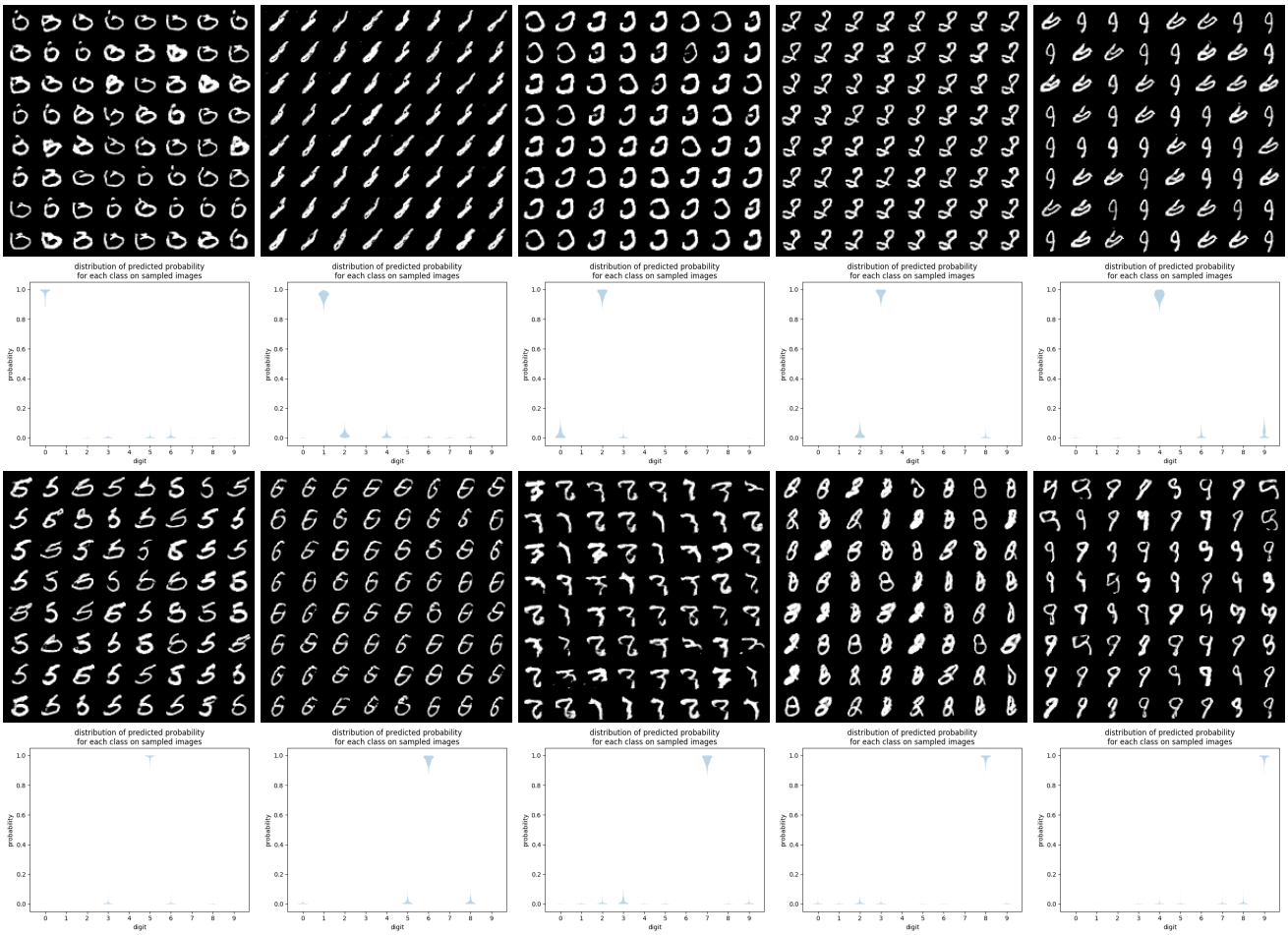


Figure 19: Examples and violin plots for high confidence misclassified examples. Top two rows: 0-4; bottom two rows: 5-9.

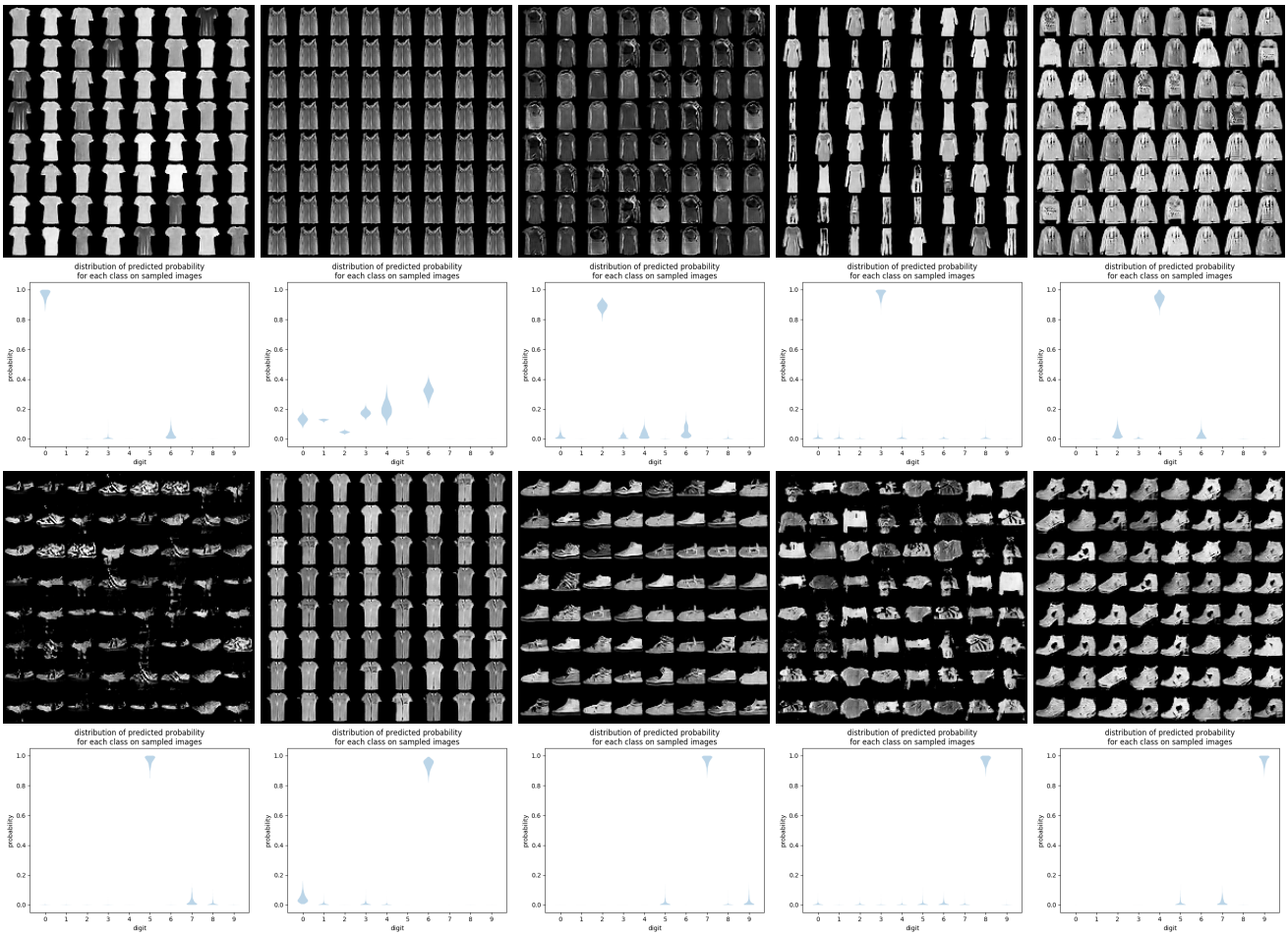


Figure 20: Samples and violin plots for high confidence misclassified examples. Top row: T-shirt, trousers (sample failure), pullover, dress, coat. Bottom row: sandal, shirt, sneaker, bag, ankle boot.

J More Details on the Novel Class Extrapolation Experiment

While training and test distributions should ideally be the same, this assumption is often violated. One such violation occurs when a model encounters data from a new class. For example, if an autonomous driving system is trained on images of cars, pedestrians, and cyclists, it is desirable to know how such a model would behave when it sees a tandem bike on the road. While many methods have been proposed to detect out-of-distribution data (Vernekar et al. 2019; DeVries and Taylor 2018; Hendrycks and Gimpel 2016; Liang, Li, and Srikant 2017), BAYES-TREX instead facilitates model insight by finding examples from novel classes that are likely to cause incorrect classifications. This analysis can help expose a model's inductive biases.

Figure 21 shows novel class extrapolation examples for CLEVR. Similar to the protocol for generating high confidence misclassified examples, for each classification target (e.g. "1 Cube"), we revoked the ability of the generator to produce objects belonging to the target class (e.g., no cubes). We also introduced a new class to the generator, a cone. For the "5 Cubes" query, we increased the stopping criterion for Random-Walk Metropolis to 1500 examples; for all other experiments, we use the standard 500 samples criterion.

(a) $P_{\text{1 Sph.}} = 99.3\%$ (b) $P_{\text{1 Sph.}} = 95.9\%$ (c) $P_{\text{1 Sph.}} = 99.3\%$ (d) $P_{\text{1 Sph.}} = 97.7\%$ (e) $P_{\text{1 Sph.}} = 97.3\%$

(f) $P_{\text{1 Cube}} = 99.2\%$ (g) $P_{\text{1 Cube}} = 97.5\%$ (h) $P_{\text{1 Cube}} = 98.7\%$ (i) $P_{\text{1 Cube}} = 99.0\%$ (j) $P_{\text{1 Cube}} = 98.7\%$

(k) $P_{\text{1 Cyl.}} = 96.9\%$ (l) $P_{\text{1 Cyl.}} = 99.1\%$ (m) $P_{\text{1 Cyl.}} = 96.5\%$ (n) $P_{\text{1 Cyl.}} = 97.2\%$ (o) $P_{\text{1 Cyl.}} = 99.0\%$

(p) $P_{\text{5 Cubes}} = 74.6\%$ (q) $P_{\text{5 Cubes}} = 89.5\%$ (r) $P_{\text{5 Cubes}} = 93.3\%$ (s) $P_{\text{5 Cubes}} = 91.6\%$ (t) $P_{\text{5 Cubes}} = 89.9\%$

Figure 21: Sampled novel class extrapolation examples and their associated prediction confidences. Similar to high confidence misclassified examples, for each target constraint (e.g., "1 Cube"), we deprive the generator of the ability to produce examples from the target class (e.g., cubes). For extrapolation, we equip the data generator with the ability to generate cones, a novel class not present in the training distribution. 21(n) is the only example which by chance does not include a novel class object.

To evaluate this extrapolation case on (Fashion-)MNIST, we consider the case of disjoint classifier and generator support as in Fig. 3(d). We split a dataset D into two disjoint parts, D_1 and D_2 by class label sets C_1 and C_2 respectively, with $C_1 \cap C_2 = \emptyset$. Then we train a classifier on D_1 and a GAN generator on D_2. We then sample high-confidence examples for each class in C_1 from the GAN-induced distribution. For MNIST, we choose $C_1 = \{0, 1, 3, 6, 9\}$, and $C_2 = \{2, 4, 5, 7, 8\}$. For Fashion-MNIST, we choose $C_1 = \{$Pullover, Dress, Sandal, Shirt, Ankle boot$\}$, and $C_2 = \{$T-shirt, Trousers, Coat, Sneaker, Bag$\}$. Fig. 7(c, d) depicts samples from a GAN trained on D_2 for which the classifier trained on D_1 has high confidence for classes in C_1.

Fig. 22 shows examples for novel-class extrapolation on MNIST. The classifier is trained on digit 0, 1, 3, 6 and 9, and tested on images generated by a GAN trained on digit 2, 4, 5, 7 and 8. Fig. 23 shows examples for novel-class extrapolation on Fashion-MNIST. The classifier is trained on pullover, dress, sandal, shirt and ankle boot, and tested on images generated by a GAN trained on T-shirt, trousers, coat, sneaker and bag.

For Fashion-MNIST, some results are intuitive. For example, the only shoe class in set C_2 is "sneaker," which accounts for most of the "sandal" and "ankle boot" predictions. The same is true for "T-shirt" $\rightarrow$ "shirt." Nevertheless, the classifier will also classify most of trousers as dresses, and bags of different styles will be classified as dresses, sandals, shirts, and ankle boots, despite visual dissimilarity. By comparison, for MNIST, it is hard to explain most of the samples. Several digits (e.g., 7) are inconsistently classified as different classes (i.e., split across 0, 1, and 9), likely because the digits are too visually distinct to allow for reasonable extrapolation.

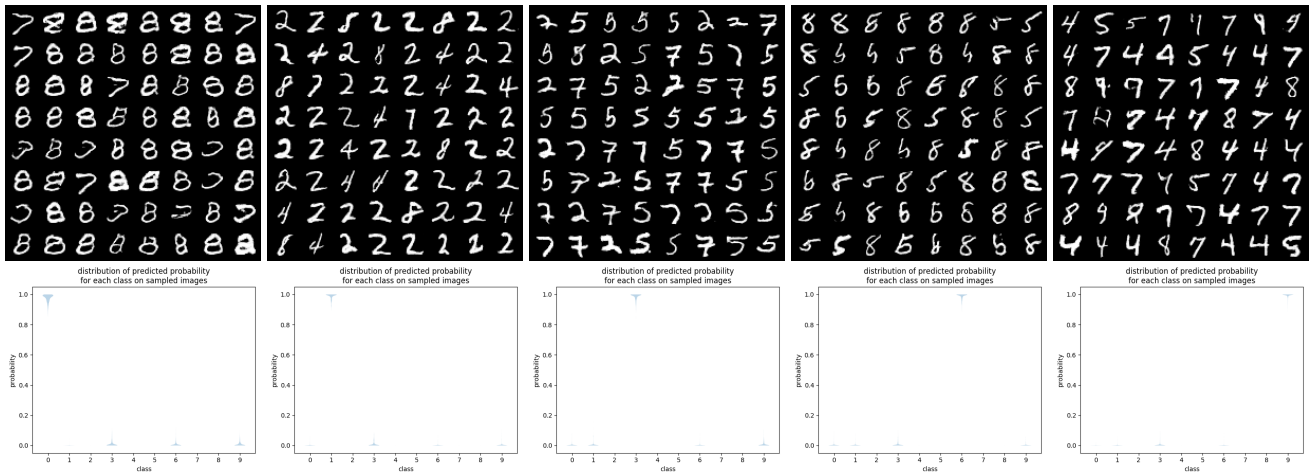


Figure 22: Samples and violin plots for novel-class extrapolation. Examples drawn from classifier classes (0, 1, 3, 6 and 9, in that order).

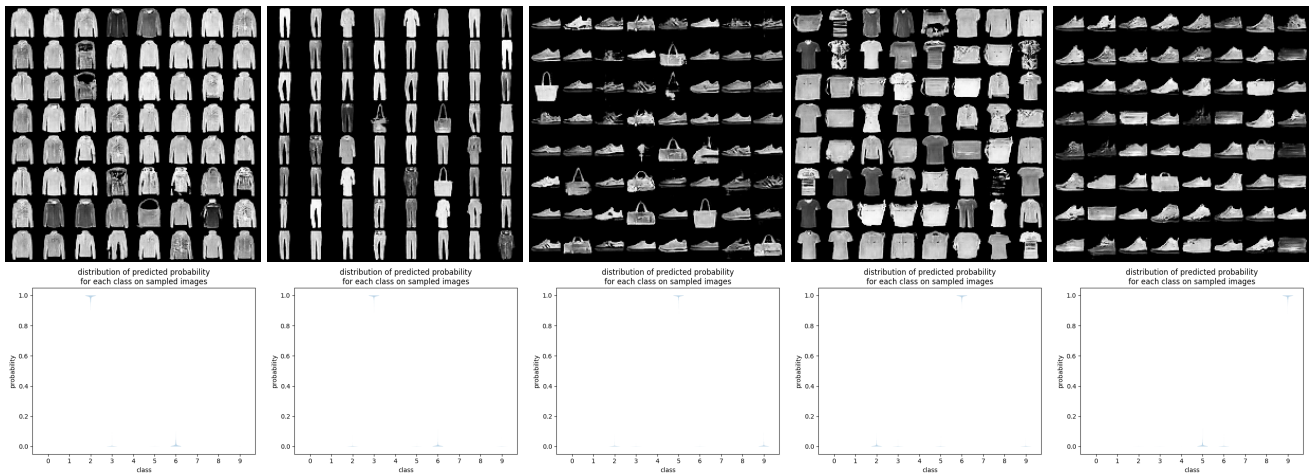


Figure 23: Samples and violin plots for novel-class extrapolation. Examples drawn from classifier classes (pullover, dress, sandal, shirt and ankle boot, in that order).

K More Details on the Domain Adaptation Experiment

In domain adaptation, we have a source domain $\mathcal{X}_S$ and a target domain $\mathcal{X}_T$, with shared label set C and cross-domain consistent decision rule $p(y|x)$ (Arjovsky et al. 2019; Long et al. 2015; Tzeng et al. 2014). The need for domain adaptation arises when the non-causal features vary across datasets. For example, for object recognition, lighting condition does not affect the object's identity, and domain adaptation techniques can be applied to learn on bright images while being tested on dark ones. Domain adaptation is especially important in robotics, where algorithms trained in simulation often fail when using real sensor data in deployment (Peng et al. 2018; Tobin et al. 2017; Tan et al. 2018).

To inspect a domain adaptation model, we sample high-confidence examples from the target domain for each class as predicted by the source-trained classifier. In our experiments, we train the adversarial discriminative domain adaptation (ADDA) model (Tzeng et al. 2017) using the code provided by the authors without modification[1], with Street View House Number (SVHN) (Netzer et al. 2011) as the source domain and MNIST as the target domain. We also trained a baseline model with the same architecture on SVHN without the domain adaptation technique. Overall, we found that the baseline model achieves 61% target accuracy on MNIST, while the ADDA model achieves 71%.

To gain further insight into the domain adapted model, we sampled MNIST images with high prediction confidence for each class. Figure 24 shows additional samples and violin plots for the baseline model in the domain adaptation analysis. Top two rows are for digit 0-4, and bottom two rows are for digit 5-9. Figure 25 shows additional samples and violin plots for the adversarial discriminative domain adaptation (ADDA) model in the domain adaptation analysis. Top two rows are for digit 0-4, and bottom two rows are for digit 5-9.

Although the average confidence for the target class is over 99%, we can see both models have several wrong predictions. For example, some digit 6s are confidently classified as 4s by the baseline model, while some 0s are classified as 1s by the ADDA model. In addition, for each model and each target digit, we performed a human labeling on 10 images, and we check how many of those images are correctly labeled. The result is summarized in Table 11. Surprisingly, on high confidence samples, the baseline model is more accurate than the ADDA model. Although this does not fundamentally contradict the improved transfer performance for ADDA, this does highlight a concerning fact: ADDA seems to suffer from the overconfidence problem more severely, as a larger proportion of high confidence samples are incorrect. This fine-grained analysis suggests the need for more extensive investigation into and potential calibration of confidence for domain adaptation models.

Table 11: Per-digit and aggregate accuracy among high confidence MNIST samples for the baseline and domain adjusted (ADDA) models. While the ADDA model has overall higher accuracy than the baseline (0.71 vs. 0.61), on high-confidence samples it has lower accuracy than the baseline (0.72 vs. 0.80), suggesting an overconfident ADDA model.

Model \ Digit	0	1	2	3	4	5	6	7	8	9	All
Baseline	1.0	0.6	1.0	0.7	0.5	0.9	0.9	0.7	1.0	0.7	0.80
ADDA	0.9	0.0	0.8	0.9	0.2	1.0	0.8	1.0	1.0	0.6	0.72

[1] https://github.com/erictzeng/adda. SVHN images are converted to gray-scale and 28×28 to match MNIST data.

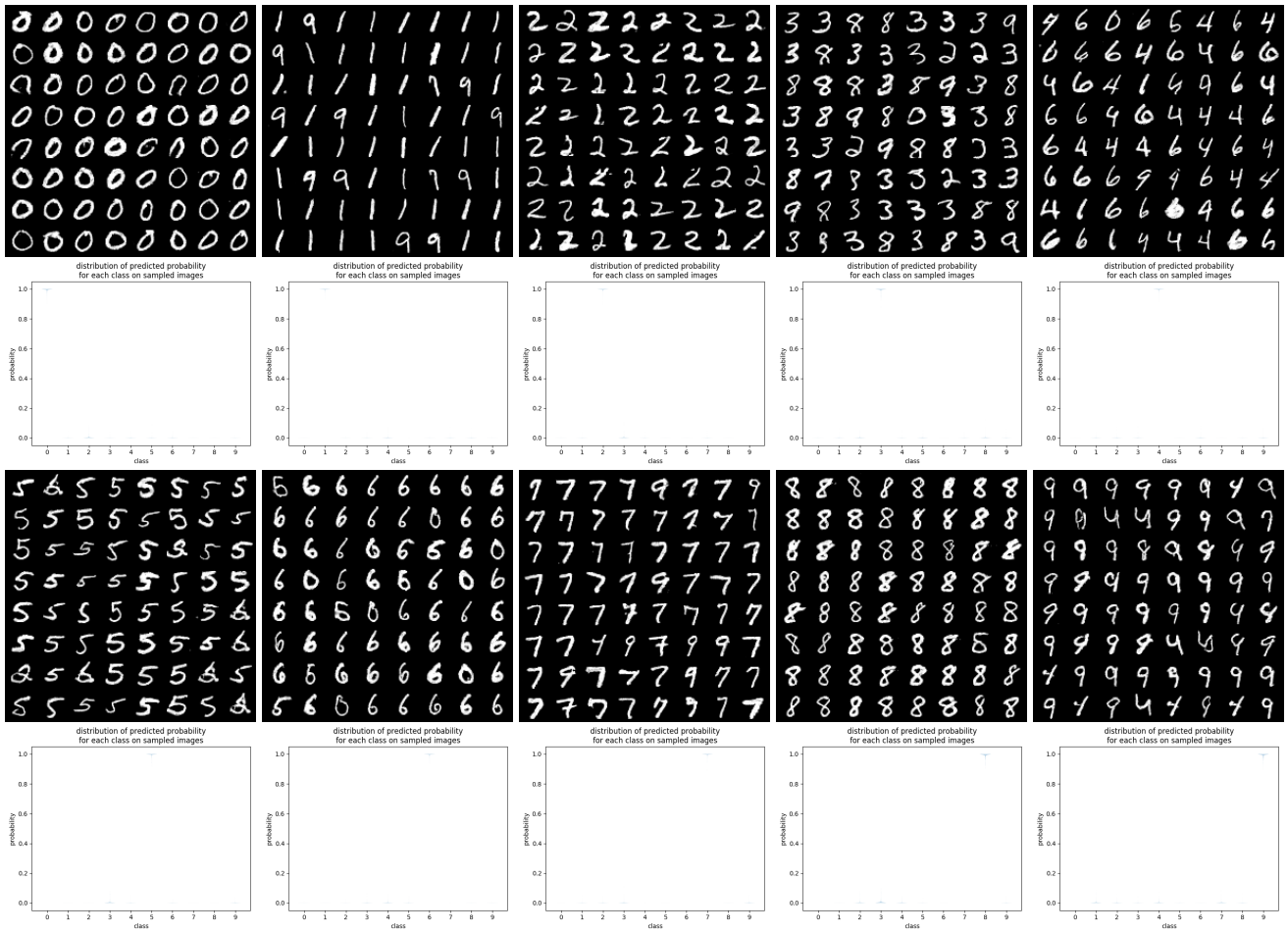


Figure 24: High confident MNIST samples generated for each class as predicted by the baseline model trained on SVHN dataset. Top row: digits 0-4. Bottom row: digits 5-9.

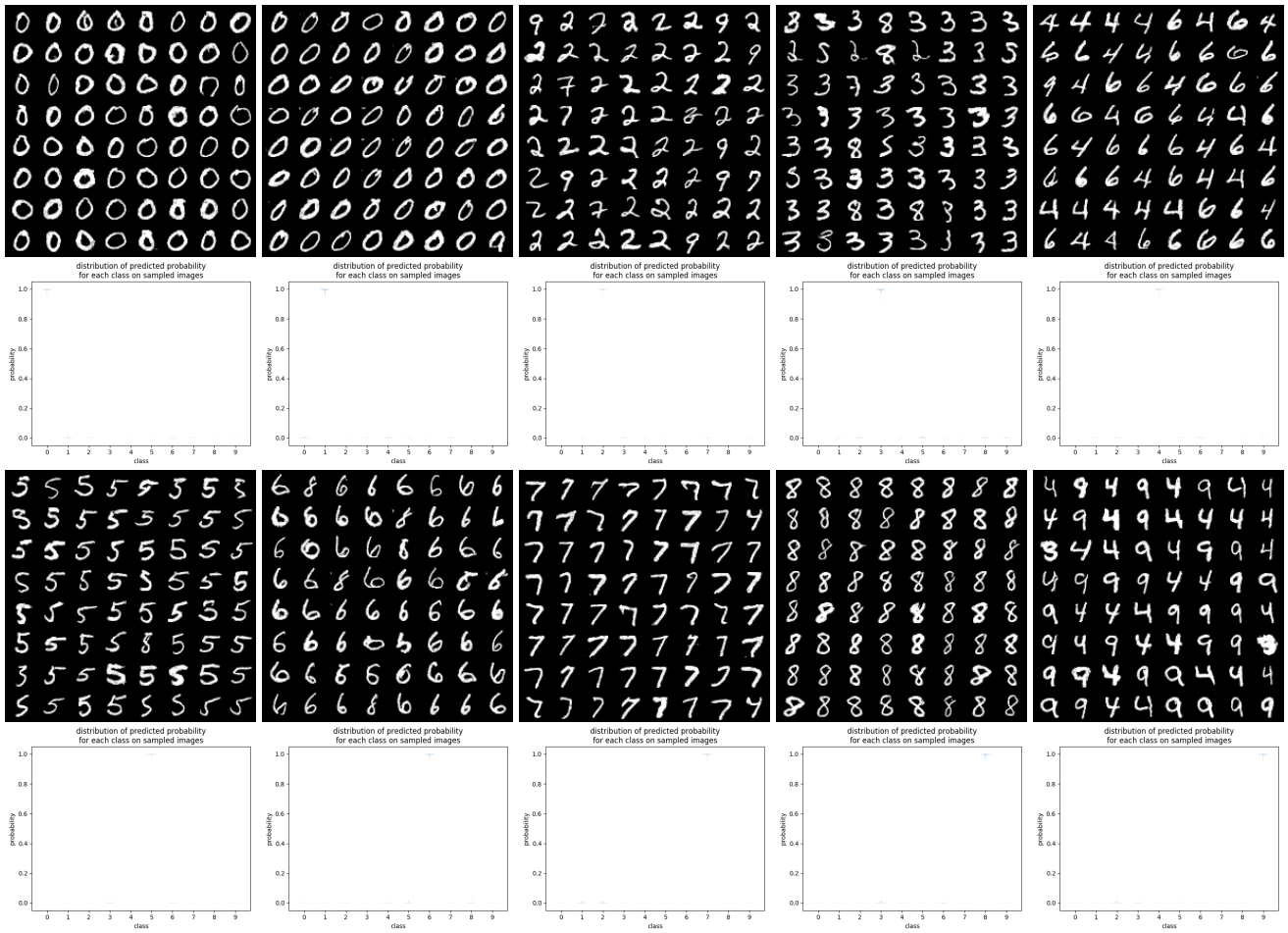


Figure 25: High confident MNIST samples generated for each class as predicted by the ADDA model trained on SVHN dataset. Top row: digits 0-4. Bottom row: digits 5-9.

L Case Study: Using BAYES-TREX with Saliency Maps

BAYES-TREX can be used to find interesting examples subject to user-specified constraints. Local interpretability techniques, such as saliency maps, can in turn be applied to these examples to derive insights. We use SmoothGrad saliency maps for our evaluation (Smilkov et al. 2017), as this technique passes the saliency map sanity checks (Adebayo et al. 2018). Fig. 28 shows several examples of CLEVR inputs and their accompanying SmoothGrad saliency maps.

L.1 High Confidence Misclassified Examples

Consider an example of a high confidence misclassified example, uncovered by BAYES-TREX. Figure 26 shows one such example and its saliency map in which the classifier has high confidence contains 1 cube though the scene is composed of only cylinders and spheres. The scene is pre-processed for input to the CLEVR classifier. This pre-processing step includes resizing the image to 224×224 pixels. The accompanying saliency map suggests that the red metal cylinder is the cause of the classifier's confusion.

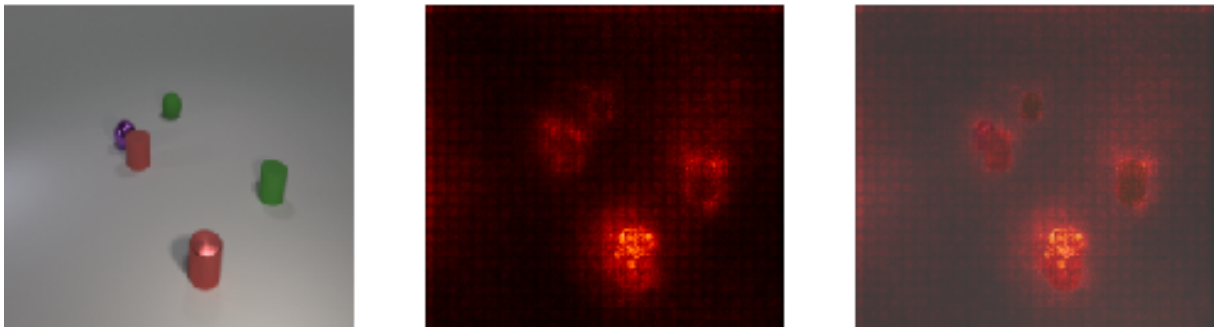
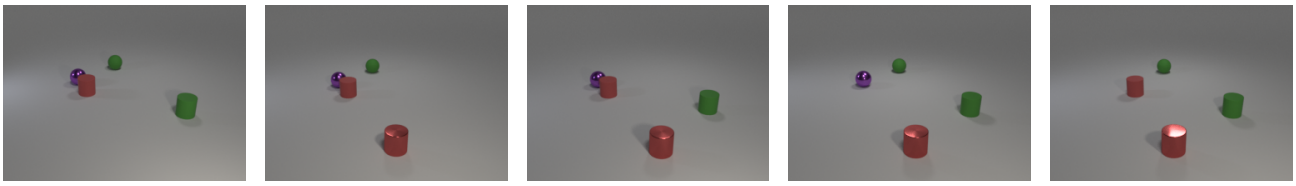


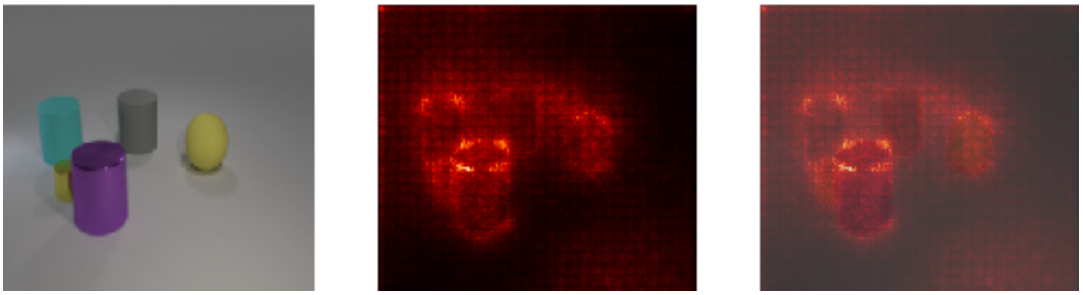
Figure 26: Left: the original image, preprocessed for classification by resizing and normalizing. The classifier is 93.5% confident this scene contains 1 cube, when in fact it is composed of 3 cylinders and 2 spheres. Middle: the SmoothGrad saliency map for this input. Right: the saliency map overlaid upon the original image. This saliency map most strongly highlights the red metal cylinder, indicating that this cylinder is likely the cause of the misclassification.

We re-render the scene, removing each object and re-classifying. As shown in Figure 27 and suggested by the original saliency map, we find the classifier mistook the red metal cylinder as a cube. Removing the red metal cylinder reduces the classifier's confidence that the scene contained 1 Cube from 93.5% to 29.0%.

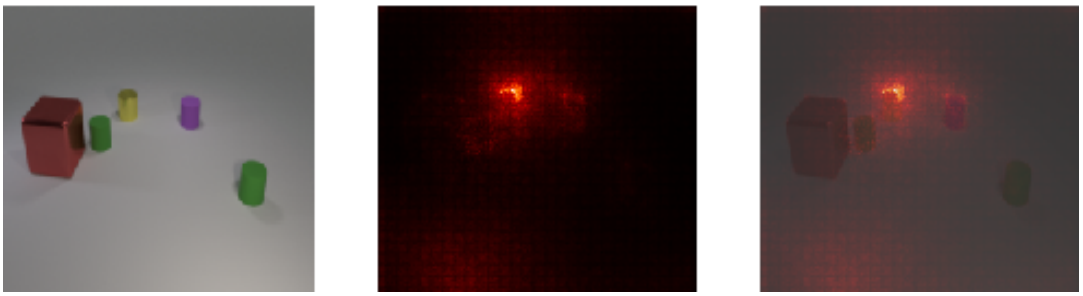


(a) $P_{\text{1 Cube}} = 29.0\%$ (b) $P_{\text{1 Cube}} = 68.5\%$ (c) $P_{\text{1 Cube}} = 81.2\%$ (d) $P_{\text{1 Cube}} = 99.0\%$ (e) $P_{\text{1 Cube}} = 99.4\%$

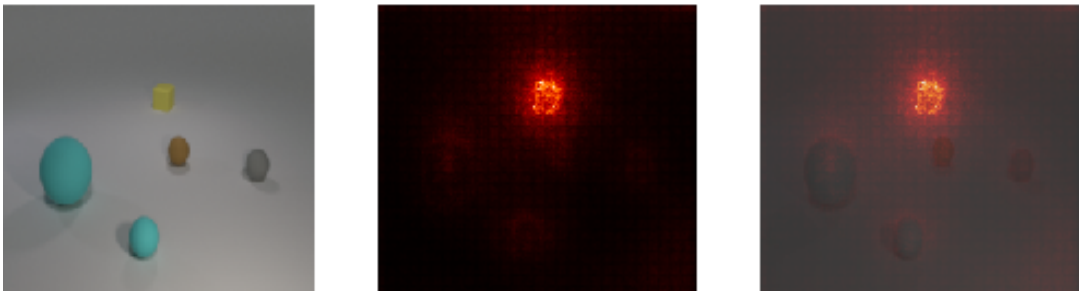
Figure 27: Re-rendering the scene, leaving out each object in turn. (a) The red metal cylinder is removed. (b) The green cylinder is removed. (c) The green sphere is removed. (d) The red rubber cylinder is removed. (e) The purple sphere is removed. As suggested by the saliency map, the removal of the red metal cylinder most prominently reduces the classification confidence, from 93.5% to 29.0%. The other scenes see a smaller confidence drop, and continue to be classified overall as containing one cube.



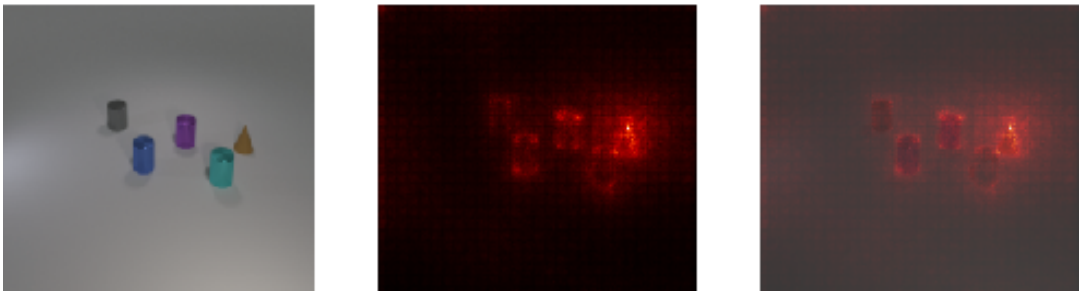
(a) Original image: $P_{\text{1 Cube}} = 85.5\%$. Purple cylinder removed: $P_{\text{1 Cube}} = 1.9\%$



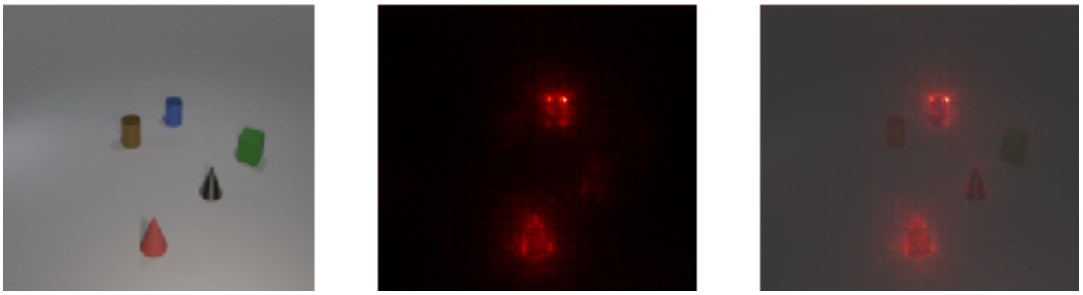
(b) Original image: $P_{\text{1 Sphere}} = 97.9\%$. Yellow cylinder removed: $P_{\text{1 Sphere}} = 5.2\%$



(c) Original image: $P_{\text{1 Cylinder}} = 85.4\%$. Red sphere removed: $P_{\text{1 Cylinder}} = 0.9\%$



(d) Original image: $P_{\text{1 Cube}} = 99.7\%$. Cone removed: $P_{\text{1 Cube}} = 0.4\%$



(e) Original image: $P_{\text{1 Sphere}} = 98.0\%$. Gray cone removed: $P_{\text{1 Sphere}} = 0.3\%$

Figure 28: Images sampled with BAYES-TREX and their saliency maps. 28(a)-28(c) are high confidence misclassified examples; 28(d)-28(e) are novel class extrapolation examples. In 28(e), the saliency map primarily highlights two objects: the red cone and the blue cylinder. Removing either of these objects does not result in a change of prediction. Instead, the misclassification of 1 Sphere is due to the marginally-highlighted gray cone.