Detecting out-of-distribution inputs using deep generative models: Pitfalls and promises

Balaji Lakshminarayanan

balajiln@

Joint work with colleagues at DeepMind and Google
Goal: How do we build neural networks that know what they don’t know?¹

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)

¹Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift [7].
Goal: How do we build neural networks that *know what they don’t know*?¹

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Dealing with train-test skew in production systems

¹Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift [7].
Goal: How do we build neural networks that *know what they don’t know*?¹

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Dealing with train-test skew in production systems
- Open-set recognition

¹Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift [7].
Goal: How do we build neural networks that know what they don’t know?¹

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Dealing with train-test skew in production systems
- Open-set recognition
- Active learning for efficient data collection

¹Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift [7].
Goal: How do we build neural networks that know what they don’t know?\(^1\)

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Dealing with train-test skew in production systems
- Open-set recognition
- Active learning for efficient data collection
- Reinforcement learning: (Safe) Exploration

---

\(^1\)Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift [7].
Goal: How do we build neural networks that know what they don’t know?¹

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Dealing with train-test skew in production systems
- Open-set recognition
- Active learning for efficient data collection
- Reinforcement learning: (Safe) Exploration
- ... and many more!

¹Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift [7].
Probabilistic Machine Learning

\[ p(y|x) \]
Discriminative vs Generative models

- $p(y|x)$ is trained only on $x \sim p_{\text{TRAIN}}(x)$
- $p(y|x)$ is typically accurate on i.i.d test inputs, but can make overconfident errors when asked to predict on out-of-distribution (OOD) inputs
- Use density model $p(x)$ to decide when to trust $p(y|x)$

“Discriminative” Model

“Generative” Model
Discriminative vs Generative models

- $p(y|x)$ is trained only on $x \sim p_{\text{TRAIN}}(x)$

\[ p(y|x) \]

“Discriminative” Model

\[ p(x) \]

“Generative” Model
Discriminative vs Generative models

- $p(y|x)$ is trained only on $x \sim p_{TRAIN}(x)$
- $p(y|x)$ is typically accurate on i.i.d test inputs, but can make overconfident errors when asked to predict on out-of-distribution (OOD) inputs
**Discriminative vs Generative models**

- $p(y|x)$ is trained only on $x \sim p_{TRAIN}(x)$
- $p(y|x)$ is typically accurate on i.i.d test inputs, but can make overconfident errors when asked to predict on out-of-distribution (OOD) inputs
- Use density model $p(x)$ to decide when to trust $p(y|x)$ [1]
Novelty Detection & Neural Network Validation

Use $p(X)$ model to reject inputs with density below some threshold [Bishop, 1994].

If $p(x^*; \phi) < \tau$, then reject $x^*$. 
Hybrids of Generative & Discriminative models

Hybrid Models with Deep and Invertible Features

Eric Nalisnick *, Akihiro Matsukawa *, Yee Whye Teh *, Dilan Gorur *, Balaji Lakshminarayanan *

• **Idea**: use normalizing flows to compute exact density $p(x)$ and $p(y|x)$ in a single feed-forward pass
Hybrids of Generative & Discriminative models

- **Idea**: use normalizing flows to compute exact density $p(x)$ and $p(y|x)$ in a single feed-forward pass
- **Works well in some cases**

Hybrid Models with Deep and Invertible Features

Eric Nalisnick * 1  Akihiro Matsukawa * 1  Yee Whye Teh 1  Dilan Gorur 1  Balaji Lakshminarayanan 1
Hybrids of Generative & Discriminative models

Hybrid Models with Deep and Invertible Features

Eric Nalisnick * 1  Akihiro Matsukawa * 1  Yee Whye Teh 1  Dilan Gorur 1  Balaji Lakshminarayanan 1

• **Idea:** use normalizing flows to compute exact density $p(x)$ and $p(y|x)$ in a single feed-forward pass
• Works well in some cases
• The failure modes were very interesting, so we decided to investigate this in detail ...
DO DEEP GENERATIVE MODELS KNOW WHAT THEY DON’T KNOW?

Eric Nalisnick†, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, Balaji Lakshminarayanan*
DeepMind
Generative models for CIFAR

Deep generative models where density $p(x)$ can be computed:

- Auto-regressive models: PixelCNNs [9]
- Variational Auto-Encoders (lower bound)
Training on CIFAR and Testing on SVHN (OOD)

Training: **CIFAR-10**

Testing: **SVHN**

\[ p(x_{CIFAR-10}) > p(x_{SVHN}) \]
Training a Flow-Based Model on CIFAR-10

CIFAR-10 Training Images

<table>
<thead>
<tr>
<th></th>
<th>CIFAR10-Train</th>
<th>CIFAR10-Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits Per Dimension</td>
<td>3.386</td>
<td>3.464</td>
</tr>
</tbody>
</table>

(Lower is Better)

(Higher is Better)
Training a Flow-Based Model on CIFAR-10

SVHN Test Images

<table>
<thead>
<tr>
<th></th>
<th>Bits Per Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR10-Train</td>
<td>3.386</td>
</tr>
<tr>
<td>CIFAR10-Test</td>
<td>3.464</td>
</tr>
<tr>
<td>SVHN-Test</td>
<td>2.389</td>
</tr>
</tbody>
</table>

(Lower is Better)

Log p(X) (Higher is Better)
Training a Flow-Based Model on CIFAR-10

Big Problem!

<table>
<thead>
<tr>
<th></th>
<th>Bits Per Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR10-Train</td>
<td>3.386</td>
</tr>
<tr>
<td>CIFAR10-Test</td>
<td>3.464</td>
</tr>
<tr>
<td>SVHN-Test</td>
<td><strong>2.389</strong></td>
</tr>
</tbody>
</table>

(Lower is Better)

SVHN Test Images
Model assigns high likelihood to constant inputs too

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Avg. Bits Per Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glow Trained on CIFAR-10</td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>15.773</td>
</tr>
<tr>
<td>Constant (128)</td>
<td>0.589</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits Per Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR10-Train</td>
</tr>
<tr>
<td>CIFAR10-Test</td>
</tr>
<tr>
<td>SVHN-Test</td>
</tr>
</tbody>
</table>

(Lower is Better)
Phenomenon holds for VAEs and PixelCNN too.
The phenomenon is asymmetric w.r.t. datasets

CIFAR-10 vs SVHN

SVHN vs CIFAR-10
Additional OOD dataset pairs

FashionMNIST vs MNIST

CelebA vs SVHN

ImageNet vs CIFAR-10 vs SVHN
Phenomenon holds throughout training

During Optimization
Ensembling does not fix the problem either

CIFAR-10 vs SVHN
1 Glow

CIFAR-10 vs SVHN
Ensemble of 10 Glows
Explaining the failure mode for Flow-based models
Define $Z$ by a transformation of another variable $X$:

$$Z = f(X)$$

Change of Variables Formula ($X \to Z$):

$$p_z(f(X)) \left| \frac{df(X)}{dX} \right| = p(X)$$
Flows: one slide summary

Define $Z$ by a transformation of another variable $X$:

$$Z = f(X)$$

$f(x)$ must be a bijection (invertible 1:1 mapping)

$x = f^{-1}(z) \quad z = f(x)$

Change of Variables Formula ($X \rightarrow Z$):

$$p_z(f(X)) \left| \frac{df(X)}{dX} \right| = p(X)$$
Flows: one slide summary

Define $Z$ by a transformation of another variable $X$:

$$ Z = f(X) $$

$f(x)$ must be a bijection (invertible 1:1 mapping)

Change of Variables Formula $(X \rightarrow Z)$:

$$ p_Z(f(X)) \left| \frac{df(X)}{dX} \right| = p(X) $$

Use simple base distribution $p_Z$ such as Gaussian

Use architecture such that determinant of Jacobian $|df/dx|$ is easy to compute

$$ x = f^{-1}(z) \quad z = f(x) $$
Define $Z$ by a transformation of another variable $X$:

$$Z = f(X)$$

$f(x)$ must be a bijection (invertible 1:1 mapping)

$x = f^{-1}(Z)$  \quad  Z = f(x)$

Change of Variables Formula ($X \rightarrow Z$):

$$p_z(f(X)) \left| \frac{df(X)}{dX} \right| = p(X)$$

Use simple base distribution $p_z$ such as Gaussian

Use architecture such that determinant of Jacobian $|df/dx|$ is easy to compute

Compose simple $f$'s to build a powerful model $f = f_1 \circ f_2 \circ \ldots \circ f_L$
When would out-of-distribution $q$ will have higher log-likelihood than $p^*$?

Mathematical characterization:

$$0 < \mathbb{E}_q[\log p(x; \theta)] - \mathbb{E}_{p^*}[\log p(x; \theta)]$$
Explaining the observations using flow models

Mathematical characterization:

\[
0 < \mathbb{E}_q[\log p(\mathbf{x}; \theta)] - \mathbb{E}_{p^*}[\log p(\mathbf{x}; \theta)]
\]

\[
\approx \frac{1}{2} \text{Tr} \left\{ \nabla_{x_0}^2 \log p_z(f(x_0; \phi)) + \nabla_{x_0}^2 \log \left| \frac{\partial f_\phi}{\partial x_0} \right| \right\} (\Sigma_q - \Sigma_{p^*})
\]

Change-of-Variable Terms

- Non-Training Distribution
- Training Distribution
- Second Moment of Training Distribution
- Second Moment of Non-Training Distribution
Explaining the observations using Constant Volume GLOW (CV GLOW)

Mathematical characterization:

$$0 < \mathbb{E}_q[\log p(\mathbf{x}; \theta)] - \mathbb{E}_{p^*}[\log p(\mathbf{x}; \theta)]$$

$$\simeq \frac{1}{2} \text{Tr} \left\{ \nabla^2_{\mathbf{x}_0} \log p_z(f(\mathbf{x}_0; \phi)) + \nabla^2_{\mathbf{x}_0} \log \left| \frac{\partial f}{\partial \mathbf{x}_0} \right| \right\} \left( \Sigma_q - \Sigma_{p^*} \right)$$

Change-of-Variable Terms
Explaining the observations using CV-GLOW

Plugging in the CV-Glow transform:

\[
\text{Tr} \left\{ \left[ \nabla^2_{\mathbf{x}_0} \log p(\mathbf{x}_0; \theta) \right] \left( \Sigma_q - \Sigma_{p^*} \right) \right\} = \sum_{c=1}^{C} \left( \prod_{k=1}^{K} \sum_{j=1}^{C} u_{k,c,j} \right)^2 \sum_{h,w} \left( \sigma^2_{q,h,w,c} - \sigma^2_{p^*,h,w,c} \right)
\]

< 0 for all log-concave densities (e.g. Gaussian)
Explaining the observations using CV-GLOW

$$0 < \mathbb{E}_q[\log p(x; \theta)] - \mathbb{E}_{p^*}[\log p(x; \theta)]$$

Non-Training Distribution

Training Distribution

Second Moment of Training Distribution

Second Moment of Non-Training Distribution
Explaining the observations using CV-GLOW

\[ 0 < \mathbb{E}_q \left[ \log p(x; \theta) \right] - \mathbb{E}_{p_*} \left[ \log p(x; \theta) \right] \]

Non-Training Distribution

Training Distribution

Second Moment of Training Distribution

CIFAR-10 vs SVHN (plugging in empirical moments)

- Asymmetry
- Uniform Inputs
- Ensembling
- Early Stopping
Explaining the observations using CV-GLOW

\[
0 < \mathbb{E}_q[\log p(x; \theta)] - \mathbb{E}_{p^*}[\log p(x; \theta)]
\]

- Non-Training Distribution
- Training Distribution
- Second Moment of Training Distribution
- Second Moment of Non-Training Distribution

- CIFAR-10 vs SVHN (plugging in empirical moments)
- Asymmetry (due to sub. being non-commutative)
- Uniform Inputs
- Ensembling
- Early Stopping
Explaining the observations using CV-GLOW

$$0 < \mathbb{E}_q[\log p(\mathbf{x}; \theta)] - \mathbb{E}_{p^*}[\log p(\mathbf{x}; \theta)]$$

Non-Training Distribution

Training Distribution

Second Moment of Training Distribution

Second Moment of Non-Training Distribution

- CIFAR-10 vs SVHN (plugging in empirical moments)
- Asymmetry (due to sub. being non-commutative)
- Uniform Inputs
- Ensembling
- Early Stopping
Explaining the observations using CV-GLOW

\[ 0 < \mathbb{E}_q[\log p(x; \theta)] - \mathbb{E}_{p^*}[\log p(x; \theta)] \]

- Non-Training Distribution
- Training Distribution
- Second Moment of Training Distribution

- CIFAR-10 vs SVHN (plugging in empirical moments)
- Asymmetry (due to sub. being non-commutative)
- Uniform Inputs (non-training 2nd moment is zero)
- Ensembling
- Early Stopping
Explaining the observations using CV-GLOW

\[ 0 < \mathbb{E}_q[\log p(x; \theta)] - \mathbb{E}_{p^*}[\log p(x; \theta)] \]

- **Non-Training Distribution**
- **Training Distribution**

\[ \approx \sum_{c=1}^{C} \left( \frac{\partial^2}{\partial z^2} \log p(z; \psi) \right) \]

\[ + \sum_{h,w} \left( \sigma^2_{q,h,w,c} - \sigma^2_{p^*,h,w,c} \right) \]

- **Second Moment of Non-Training Distribution**

- **CIFAR-10 vs SVHN** (plugging in empirical moments)
- **Asymmetry** (due to sub. being non-commutative)
- **Uniform Inputs** (non-training 2nd moment is zero)
- **Ensembling**
- **Early Stopping** (sign doesn’t depend on model param. values)
Explaining the observations using CV-GLOW

\[ 0 < \mathbb{E}_q[\log p(x; \theta)] - \mathbb{E}_{p^*}[\log p(x; \theta)] \]

- **Non-Training Distribution**
- **Training Distribution**
- Second Moment of Training Distribution
- Second Moment of Non-Training Distribution

- CIFAR-10 vs SVHN (plugging in empirical moments)
- Asymmetry (due to sub. being non-commutative)
- Uniform Inputs (non-training 2nd moment is zero)
- Ensembling
- Early Stopping (sign doesn’t depend on model param. values)
Explaining the observations using CV-GLOW

\[
0 < \mathbb{E}_q[\log p(x; \theta)] - \mathbb{E}_{p^*}[\log p(x; \theta)]
\]

Hypothesis: If the second-order statistics do indeed dominate, we should be able to control the likelihoods by graying the images...
Explaining the observations using CV-GLOW

$$0 < \mathbb{E}_q[\log p(\mathbf{x}; \theta)] - \mathbb{E}_{p^*}[\log p(\mathbf{x}; \theta)]$$

One weird trick to increase your likelihoods!
Follow-up Work
Detecting Out-of-Distribution Inputs to Deep Generative Models Using a Test for Typicality

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Balaji Lakshminarayanan
DeepMind
{enalisnick, amatsukawa, ywteh, balajiln}@google.com
Motivating question: why don’t we ever see samples from the OOD set?

- **FashionMNIST:**
  - Training Set

- **MNIST:**
  - Higher Likelihood

- **Samples from Generative Model**
Typical sets versus Mode

• Mode can be very atypical of the distribution in high dimensions
Typical sets versus Mode

- Mode can be very atypical of the distribution in high dimensions
- High-dimensional Gaussian:
  - Mode is at $\mu$
  - Typical samples lie near the shell

Figure: High dimensional Gaussian
Could similar phenomenon happen with deep generative models too?

High Density

High Probability (Samples)
Definition of typical sets

**Definition 2.1.** $\epsilon$-Typical Set [11]  
For a distribution $p(x)$ with support $x \in \mathcal{X}$, the $\epsilon$-typical set $\mathcal{A}_\epsilon^N[p(x)] \subseteq \mathcal{X}^N$ is comprised of all $N$-length sequences that satisfy

$$ -\frac{1}{N} \sum_{n=1}^{N} \log p(x_n) \leq H[p(x)] + \epsilon $$

where $H[p(x)] = \int \mathcal{X} p(x) [-\log p(x)] dx$ and $\epsilon \in \mathbb{R}^+$ is a small constant.
Definition of typical sets

**Definition 2.1. $\epsilon$-Typical Set [11]** For a distribution $p(x)$ with support $x \in \mathcal{X}$, the $\epsilon$-typical set $\mathcal{A}_\epsilon^N[p(x)] \in \mathcal{X}^N$ is comprised of all $N$-length sequences that satisfy

$$\mathbb{H}[p(x)] - \epsilon \leq \frac{-1}{N} \sum_{n=1}^{N} \log p(x_n) \leq \mathbb{H}[p(x)] + \epsilon$$

where $\mathbb{H}[p(x)] = \int_{\mathcal{X}} p(x) [-\log p(x)] dx$ and $\epsilon \in \mathbb{R}^+$ is a small constant.

Testing for typicality

- If a batch $x_1, \ldots, x_M$ is in the typical set, then the average negative log likelihood should be close to the entropy.
- Can use tools from statistical hypothesis testing literature
Testing for Typicality improves OOD detection

**Figure:** Effect of batch size on AUC of OOD detection
Better OOD detection for genomic sequences

Likelihood Ratios for Out-of-Distribution Detection

Jie Ren*†
Google Research
jjren@google.com

Peter J. Liu
Google Research
peterjliu@google.com

Emily Fertig†
Google Research
emilyaf@google.com

Jasper Snoek
Google Research
jsnoek@google.com

Ryan Poplin
Google Inc.
rpoplin@google.com

Mark A. Depristo
Google Inc.
mdepristo@google.com

Joshua V. Dillon
Google Research
jvdillon@google.com

Balaji Lakshminarayanan*
DeepMind
balajiln@google.com
Explaining the failure mode for PixelCNN

- PixelCNN++ model trained on FashionMNIST
- Heat-map showing per-pixel contributions on Fashion-MNIST (in-dist) and MNIST (OOD)
- **Background pixels dominate the likelihood**

\[
\log p_\theta(x_d)
\]

\[
\log p_\theta(x_d) - \log p_{\theta_0}(x_d)
\]
Explaining the failure mode for PixelCNN

- PixelCNN++ model trained on FashionMNIST
- Heat-map showing per-pixel contributions on Fashion-MNIST (in-dist) and MNIST (OOD)
- **Background pixels dominate the likelihood.** Explains why MNIST is assigned higher likelihood.

\[
\log p_\theta(x_d)
\]
Likelihood Ratio to distinguish Background vs Semantics

- Input $\mathbf{x}$ consists of background $\mathbf{x}_B$ and semantic component $\mathbf{x}_S$. Examples:
  - Images: background versus objects
  - Text: stop words versus key words
  - Genomics: GC background versus motifs
  - Speech: background noise versus speaker

$$p(\mathbf{x}) = p(\mathbf{x}_B)p(\mathbf{x}_S)$$

can be dominant
the focus
Likelihood Ratio to distinguish Background vs Semantics

• Input $x$ consists of *background* $x_B$ and semantic component $x_S$. Examples:
  – Images: background versus objects
  – Text: stop words versus key words
  – Genomics: GC background versus motifs
  – Speech: background noise versus speaker

$$p(x) = p(x_B) p(x_S)$$

• Training a background model on perturbed inputs. Compute the likelihood ratio

$$LLR(x) = \log \frac{p_\theta(x)}{p_{\theta_0}(x)} = \log \frac{p_\theta(x_B)}{p_{\theta_0}(x_B)} \frac{p_\theta(x_S)}{p_{\theta_0}(x_S)} \approx \log \frac{p_\theta(x_S)}{p_{\theta_0}(x_S)}$$
Likelihood ratio improves OOD detection for PixelCNN

- PixelCNN++ model trained on FashionMNIST
- Heat-map showing per-pixel contributions on Fashion-MNIST (in-dist) and MNIST (OOD)
- Likelihood Ratio (using background model) focuses on the semantic pixels and significantly outperforms likelihood on OOD detection.

\[
\log p_\theta(x_d) \quad \text{and} \quad \log p_\theta(x_d) - \log p_{\theta_0}(x_d)
\]
Likelihood ratio significantly improves OOD detection on genomics data too

<table>
<thead>
<tr>
<th>Method</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood</td>
<td>0.630</td>
</tr>
<tr>
<td><strong>Likelihood Ratio</strong></td>
<td><strong>0.755</strong></td>
</tr>
<tr>
<td>Classifier-based $p(y</td>
<td>x)$</td>
</tr>
<tr>
<td>Classifier-based Entropy</td>
<td>0.622</td>
</tr>
<tr>
<td>Classifier-based ODIN</td>
<td>0.645</td>
</tr>
<tr>
<td>Classifier Ensemble 5</td>
<td>0.673</td>
</tr>
<tr>
<td>Classifier-based Mahalanobis Distance</td>
<td>0.496</td>
</tr>
</tbody>
</table>

- Realistic benchmark + open-source code
- https://github.com/google-research/google-research/tree/master/genomics_ood
Take home messages

- Be cautious when using density estimates from deep generative models as proxy for “similarity” to training data
  - Can assign higher density to OOD inputs than training data!
  - Novelty / Anomaly detection
Take home messages

• Be cautious when using density estimates from deep generative models as proxy for “similarity” to training data
  – Can assign higher density to OOD inputs than training data!
  – Novelty / Anomaly detection

• Explaining the observed failure modes:
  – Flow-based models: Can be explained through inductive bias and the relative variances of the input distributions
  – Autoregressive models: Can be explained through background effect
Take home messages

• Be cautious when using density estimates from deep generative models as proxy for “similarity” to training data
  – Can assign higher density to OOD inputs than training data!
  – Novelty / Anomaly detection

• Explaining the observed failure modes:
  – Flow-based models: Can be explained through inductive bias and the relative variances of the input distributions
  – Autoregressive models: Can be explained through background effect

• Solutions:
  – Likelihood ratio using background model
  – Typicality test
Thanks!

- Aki Matsukawa
- Dilan Gorur
- Emily Fertig
- Eric Nalisnick
- Jasper Snoek
- Jie Ren
- Josh Dillon
- Mark DePristo
- Peter Liu
- Ryan Poplin
- Yee Whye Teh
Papers available on my webpage (link)

Out-of-distribution robustness of deep generative models

• *Do deep generative models know what they don’t know?* [5]
• *Likelihood ratios for out-of-distribution detection* [8]
• *Detecting out-of-distribution inputs to deep generative models using a test for typicality* [4]

Predictive uncertainty estimation in deep learning

• *Hybrid models with deep and invertible features* [6]
• *Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift* [7]
• *Simple and scalable predictive uncertainty estimation using deep ensembles* [3]


