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

**Balaji Lakshminarayanan**

**balajiln@**

Joint work with colleagues at DeepMind and Google

DeepMind

# 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)

[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

[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

[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

[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

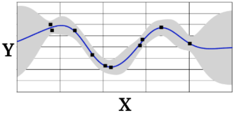
[1] Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift [7].

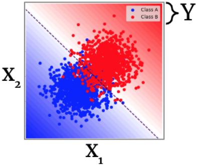
# Goal: How do we build neural networks that *know what they don't know*?[1]

- 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!

[1] Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift [7].

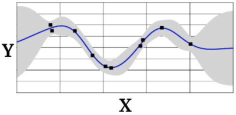
# Probabilistic Machine Learning

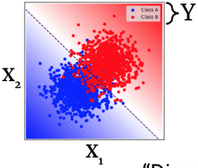



$$p(\mathbf{y}|\mathbf{x})$$

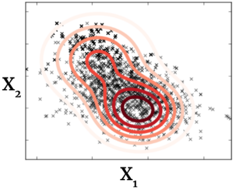
# Discriminative vs Generative models




$$p(\mathbf{y}|\mathbf{x})$$

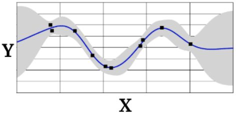
"Discriminative" Model

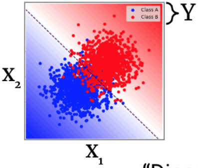



$$p(\mathbf{x})$$

"Generative" Model

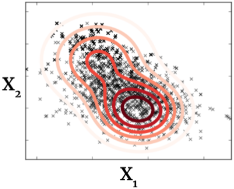
# Discriminative vs Generative models




$$p(\mathbf{y}|\mathbf{x})$$

"Discriminative" Model

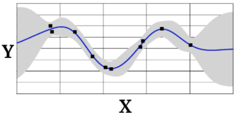


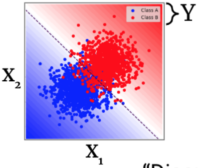

$$p(\mathbf{x})$$

"Generative" Model

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

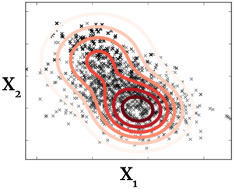
# Discriminative vs Generative models




$$p(\mathbf{y}|\mathbf{x})$$

"Discriminative" Model

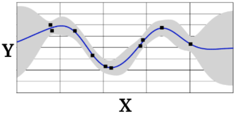


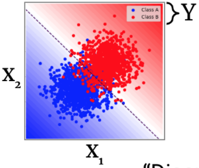

$$p(\mathbf{x})$$

"Generative" Model

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

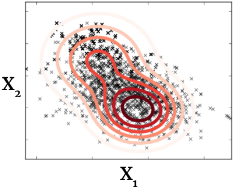
# Discriminative vs Generative models




$$p(\mathbf{y}|\mathbf{x})$$

"Discriminative" Model

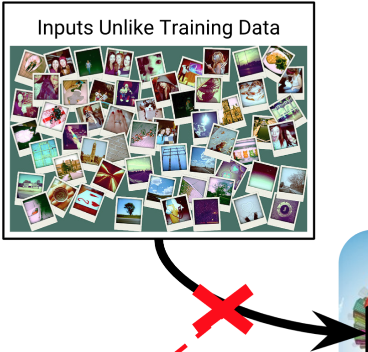



$$p(\mathbf{x})$$

"Generative" Model

- $p(y|\boldsymbol{x})$ is trained only on $x \sim p_{TRAIN}(x)$
- $p(y|\boldsymbol{x})$ 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 $p(\boldsymbol{x})$ to decide when to trust $p(y|\boldsymbol{x})$ [1]

# Novelty Detection & Neural Network Validation




$$\text{if } p(\boldsymbol{x}^*; \phi) < \tau, \quad \text{then } \texttt{reject } \boldsymbol{x}^*$$


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

# 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 $p(\boldsymbol{x})$ and $p(y|\boldsymbol{x})$ in a single feed-forward pass

# 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 $p(\boldsymbol{x})$ and $p(y|\boldsymbol{x})$ in a single feed-forward pass
- Works well in some cases

# 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]

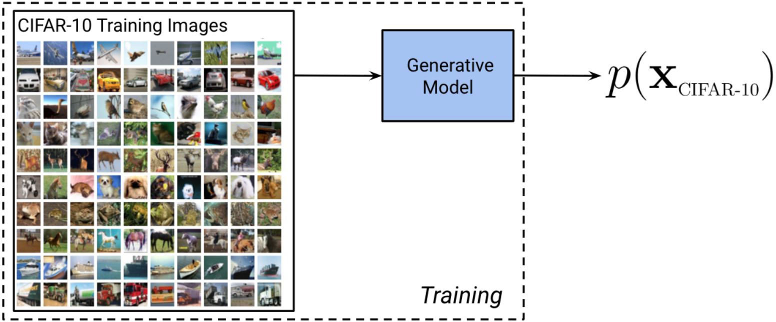
- **Idea**: use normalizing flows to compute exact density $p(\boldsymbol{x})$ and $p(y|\boldsymbol{x})$ in a single feed-forward pass
- Works well in some cases
- The failure modes were very interesting, so we decided to investigate this in detail ...

Published as a conference paper at ICLR 2019

# DO DEEP GENERATIVE MODELS KNOW WHAT THEY DON'T KNOW?

**Eric Nalisnick[*†], Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, Balaji Lakshminarayanan[*]**
DeepMind

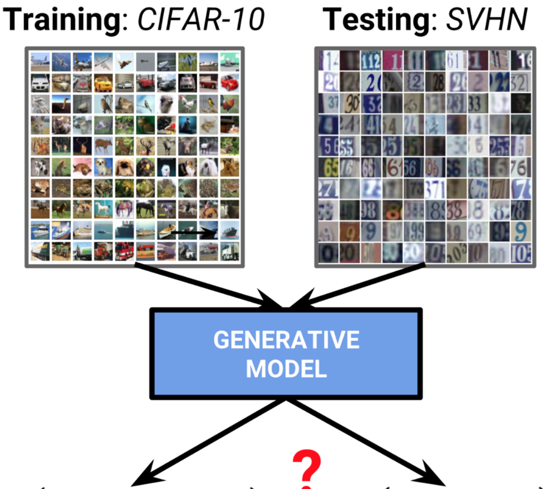
# Generative models for CIFAR




Deep generative models where density $p(\boldsymbol{x})$ can be computed:

- Flow-based models: GLOW [2]
- Auto-regressive models: PixelCNNs [9]
- Variational Auto-Encoders (lower bound)

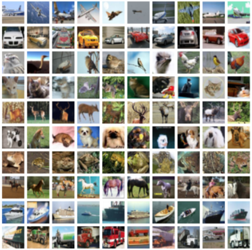
# Training on CIFAR and Testing on SVHN (OOD)




$$p(\mathbf{x}_{\text{CIFAR-10}}) \overset{?}{>} p(\mathbf{x}_{\text{SVHN}})$$

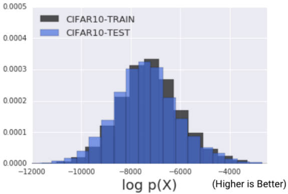
# Training a Flow-Based Model on CIFAR-10

CIFAR-10 Training Images



| | Bits Per Dimension (NLL / # dims / log 2) |
|---|---|
| CIFAR10-Train | 3.386 |
| CIFAR10-Test | 3.464 |

(Lower is Better)

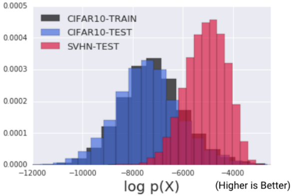


# Training a Flow-Based Model on CIFAR-10

SVHN Test Images

| | Bits Per Dimension (NLL / # dims / log 2) |
|---|---|
| CIFAR10-Train | 3.386 |
| CIFAR10-Test | 3.464 |
| SVHN-Test | **2.389** |

(Lower is Better)



# Training a Flow-Based Model on CIFAR-10

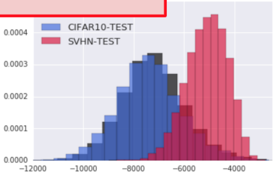
SVHN Test Images

Bits Per Dimension
(NLL / # dims / log 2)

| | |
|---|---|
| CIFAR10-Train | 3.386 |
| [illegible] | 3.464 |
| [illegible] | **2.389** |

(Lower is Better)

Big Problem!

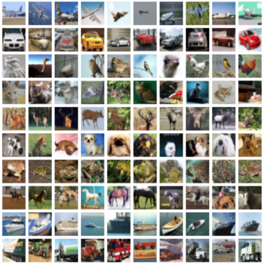



log p(X)

(Higher is Better)

# Model assigns high likelihood to constant inputs too

CIFAR-10 Training Images

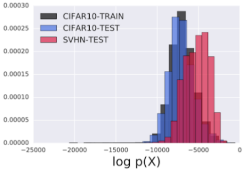


| | Bits Per Dimension (NLL / # dims / log 2) |
|---|---|
| CIFAR10-Train | 3.386 |
| CIFAR10-Test | 3.464 |
| SVHN-Test | **2.389** |

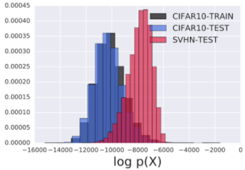
(Lower is Better)

| Data Set | Avg. Bits Per Dimension |
|---|---|
| *Glow Trained on CIFAR-10* | |
| Random | 15.773 |
| Constant (128) | **0.589** |

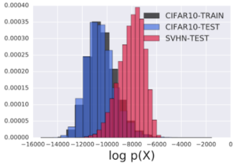
# Phenomenon holds for VAEs and PixelCNN too




(a) PixelCNN

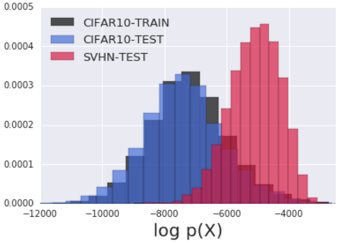



(b) VAE with RNVP as encoder

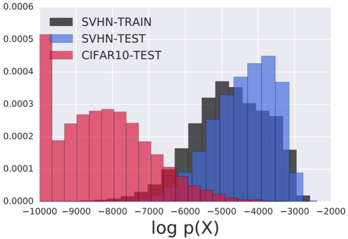



(c) VAE conv-categorical likelihood

# The phenomenon is asymmetric w.r.t. datasets

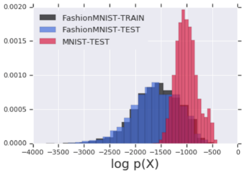



**CIFAR-10** vs **SVHN**

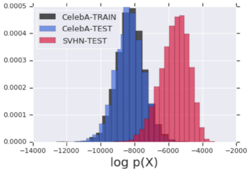



**SVHN** vs **CIFAR-10**

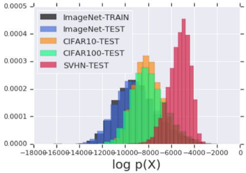
# Additional OOD dataset pairs




**FashionMNIST** vs **MNIST**

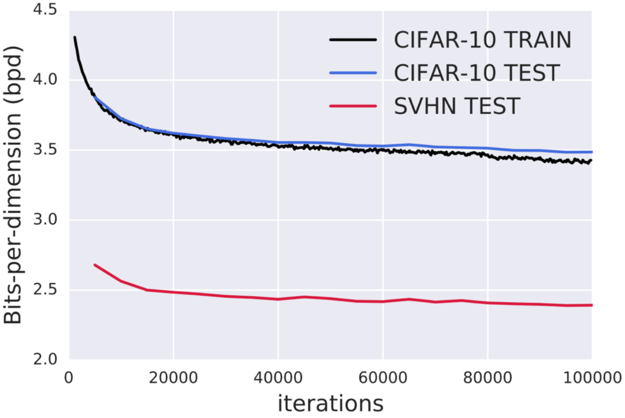



**CelebA** vs **SVHN**



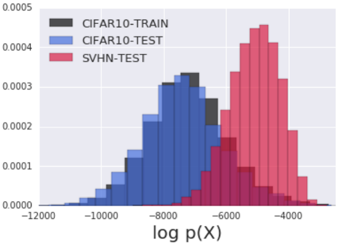

**ImageNet** vs **CIFAR-10** vs **SVHN**

# Phenomenon holds throughout training

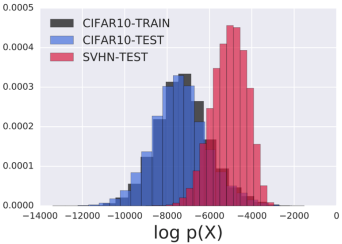



**During Optimization**

# 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

# Flows: one slide summary

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

$$Z = f(X)$$

Change of Variables Formula (X → Z):

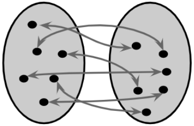
$$p_z(f(X)) \left| \frac{df(X)}{dX} \right| = p(X)$$

# 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)*



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

Change of Variables Formula (X → Z):

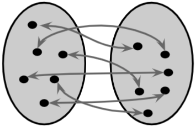
$$p_z(f(X)) \left| \frac{df(X)}{dX} \right| = p(X)$$

# Flows: one slide summary

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

$$Z = f(X)$$

$f(\mathbf{x})$ must be a *bijection (invertible 1:1 mapping)*



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

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

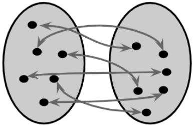
Use architecture such that determinant of Jacobian |df/dx| is easy to compute

# 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)*



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

Change of Variables Formula (X → 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:

$$0 < \underbrace{\mathbb{E}_q}_{\text{Non-Training Distribution}}[\log p(\boldsymbol{x}; \boldsymbol{\theta})] - \underbrace{\mathbb{E}_{p^*}}_{\text{Training Distribution}}[\log p(\boldsymbol{x}; \boldsymbol{\theta})]$$

# Explaining the observations using flow models

Mathematical characterization:

$$0 < \underbrace{\mathbb{E}_q}_{\text{Non-Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \underbrace{\mathbb{E}_{p^*}}_{\text{Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

$$\approx \frac{1}{2}\operatorname{Tr}\left\{\left[\underbrace{\nabla^2_{\boldsymbol{x}_0}\log p_z(f(\boldsymbol{x}_0;\boldsymbol{\phi})) + \nabla^2_{\boldsymbol{x}_0}\log\left|\frac{\partial \boldsymbol{f}_{\phi}}{\partial \boldsymbol{x}_0}\right|}_{\text{Change-of-Variable Terms}}\right]\left(\underbrace{\boldsymbol{\Sigma}_q}_{\text{Second Moment of Non-Training Distribution}} - \overbrace{\boldsymbol{\Sigma}_{p^*}}^{\text{Second Moment of Training Distribution}}\right)\right\}$$

# Explaining the observations using Constant Volume GLOW (CV GLOW)

Mathematical characterization:

$$0 < \underbrace{\mathbb{E}_q}_{\text{Non-Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \underbrace{\mathbb{E}_{p^*}}_{\text{Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

$$\approx \frac{1}{2} \operatorname{Tr}\left\{ \underbrace{\left[ \nabla^2_{\boldsymbol{x}_0} \log p_z(f(\boldsymbol{x}_0;\boldsymbol{\phi})) + \cancel{\nabla^2_{\boldsymbol{x}_0} \log \left| \frac{\partial \boldsymbol{f}_\phi}{\partial \boldsymbol{x}_0} \right|} \right]}_{\text{Change-of-Variable Terms}} \left( \underbrace{\boldsymbol{\Sigma}_q}_{\text{Second Moment of Non-Training Distribution}} - \overbrace{\boldsymbol{\Sigma}_{p^*}}^{\text{Second Moment of Training Distribution}} \right) \right\}$$

# Explaining the observations using CV-GLOW

Plugging in the CV-Glow transform:

$$\mathrm{Tr}\left\{\left[\nabla^2_{\boldsymbol{x}_0} \log p(\boldsymbol{x}_0; \boldsymbol{\theta})\right] (\boldsymbol{\Sigma}_q - \boldsymbol{\Sigma}_{p^*})\right\}$$

$$= \underbrace{\frac{\partial^2}{\partial z^2} \log p(\boldsymbol{z}; \boldsymbol{\psi})}_{\text{< 0 for all log-concave densities (e.g. Gaussian)}} \sum_{c=1}^{C} \underbrace{\left(\prod_{k=1}^{K} \sum_{j=1}^{C} u_{k,c,j}\right)^2}_{\text{Non-negative due to square}} \sum_{h,w} (\sigma^2_{q,h,w,c} - \sigma^2_{p^*,h,w,c})$$

**< 0 for all log-concave densities (e.g. Gaussian)**

**Non-negative due to square**

**Second Moment of Non-Training Distribution**

**Second Moment of Training Distribution**

# Explaining the observations using CV-GLOW

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

Non-Training Distribution

Training Distribution

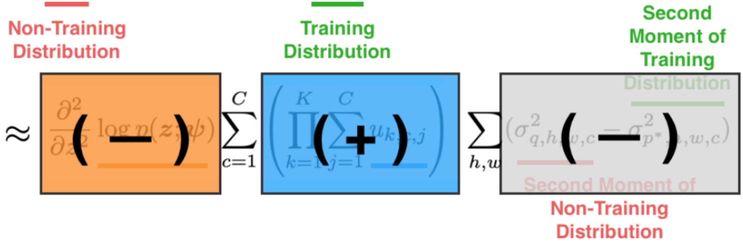
Second Moment of Training Distribution

$$\approx \boxed{(\,-\,)} \sum_{c=1}^{C} \boxed{(\,+\,)} \sum_{h,w} (\sigma^2_{q,h,w,c} - \sigma^2_{p^*,h,w,c})$$

Second Moment of Non-Training Distribution

# Explaining the observations using CV-GLOW

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




- [ ] **CIFAR-10 vs SVHN** (plugging in empirical moments)
- [ ] Asymmetry
- [ ] Uniform Inputs
- [ ] Ensembling
- [ ] Early Stopping

# Explaining the observations using CV-GLOW

$$0 < \underbrace{\mathbb{E}_q}_{\text{Non-Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \underbrace{\mathbb{E}_{p^*}}_{\text{Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

$$\approx \boxed{\underbrace{\left(\frac{\partial^2}{\partial z^2}\log p(z;\psi)\right)}_{(-)}} \sum_{c=1}^{C} \boxed{\underbrace{\left(\prod_{k=1}^{K}\sum_{j=1}^{C} u_{k,c,j}\right)}_{(+)}} \sum_{h,w}(\underbrace{\sigma^2_{q,h,w,c}}_{\text{Second Moment of Non-Training Distribution}} - \overbrace{\sigma^2_{p^*,h,w,c}}^{\text{Second Moment of Training Distribution}})$$

- [ ] **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

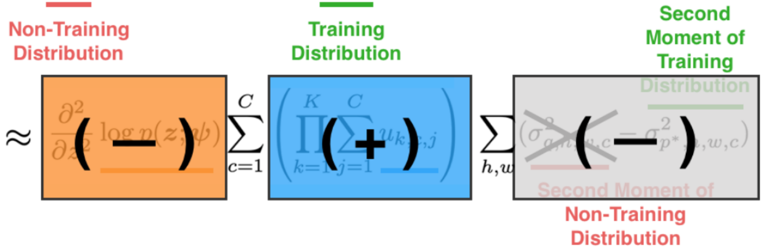
$$0 < \underbrace{\mathbb{E}_q}_{\text{Non-Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \underbrace{\mathbb{E}_{p^*}}_{\text{Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

$$\approx \boxed{(\;-\;)} \sum_{c=1}^{C} \boxed{(\;+\;)} \sum_{h,w} \Big( \underbrace{\cancel{\sigma^2_{q,h,w,c}}}_{\text{Second Moment of Non-Training Distribution}} - \overbrace{\sigma^2_{p^*,h,w,c}}^{\text{Second Moment of Training Distribution}} \Big)$$

- ☐ **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

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




- [ ] **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

$$0 < \underbrace{\mathbb{E}_q}_{\text{Non-Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \underbrace{\mathbb{E}_{p^*}}_{\text{Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

$$\approx \boxed{\left(\;-\;\right)} \sum_{c=1}^{C} \boxed{\left(\;+\;\right)} \sum_{h,w} \Big( \underbrace{\sigma^2_{q,h,w,c}}_{\text{Second Moment of Non-Training Distribution}} - \overbrace{\sigma^2_{p^*,h,w,c}}^{\text{Second Moment of Training Distribution}} \Big)$$

- ☐ **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

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

Non-Training Distribution

Training Distribution

Second Moment of Training Distribution

$$\approx \boxed{(-)} \sum_{c=1}^{C} \boxed{(+)} \sum_{h,w} (\sigma^2_{q,h,w,c} - \sigma^2_{p^*,h,w,c})$$

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

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

Non-Training Distribution

Training Distribution

Second Moment of Training Distribution

$$\approx \boxed{(\,-\,)} \sum_{c=1}^{C} \boxed{(\,+\,)} \sum_{h,w} (\sigma^2_{q,h,w,c} - \sigma^2_{p^*,h,w,c})$$

Second Moment of Non-Training Distribution

**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

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

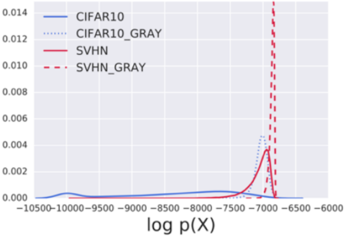
Non-Training Distribution

Training Distribution

Second Moment of Training Distribution

$$\approx \boxed{(-)} \sum_{c=1}^{C} \boxed{(+)} \sum_{h,w}(\sigma^2_{q,h,w,c} - \sigma^2_{p^*,h,w,c})$$

Second Moment of Non-Training Distribution




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

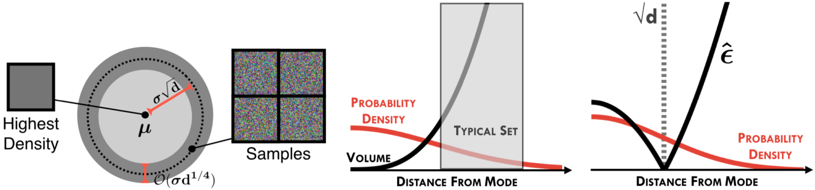
Samples from
Generative Model

# Typical sets versus Mode

- Mode can be very atypical of the distribution in high dimensions

# Typical sets versus Mode

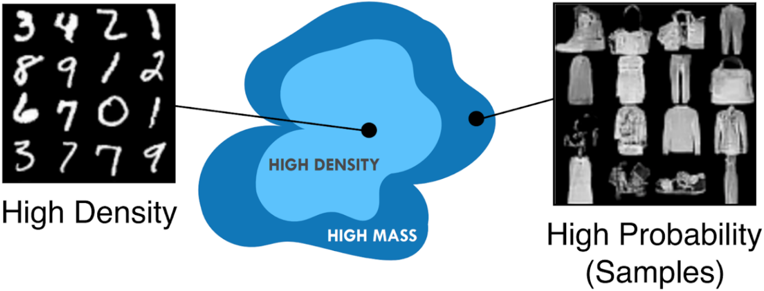
- Mode can be very atypical of the distribution in high dimensions
- High-dimensional Gaussian:
  - Mode is at $\mu$
  - Typical samples lie near the shell




(a) Gaussian Example (b) Illustration (c) Simulation

Figure: High dimensional Gaussian

# Could similar phenomenon happen with deep generative models too?



# Definition of typical sets

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

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

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

# Definition of typical sets

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

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

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

## Testing for typicality

- If a batch $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ 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

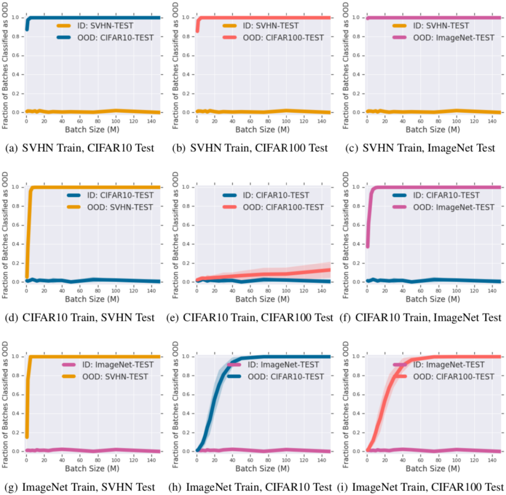



Figure: Effect of batch size on AUC of OOD detection

# 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

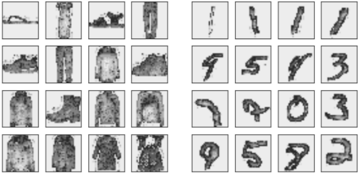
**Balaji Lakshminarayanan**[*]
DeepMind
balajiln@google.com

# 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**

# 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.*



$\log p_\theta(x_d)$


$\log p_\theta(x_d) - \log p_{\theta_0}(x_d)$

# Likelihood Ratio to distinguish Background vs Semantics

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

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

can be dominant

the focus

# Likelihood Ratio to distinguish Background vs Semantics

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

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

can be dominant

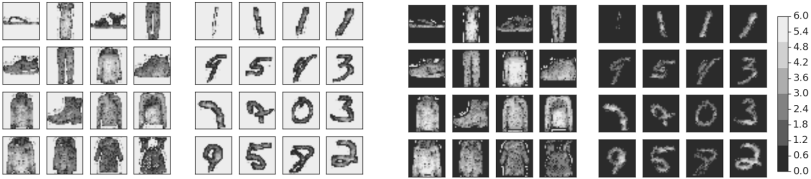
the focus

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

$$\mathsf{LLR}(\mathbf{x}) = \log \frac{p_\theta(\mathbf{x})}{p_{\theta_0}(\mathbf{x})} = \log \frac{\cancel{p_\theta(\mathbf{x}_B)}\ p_\theta(\mathbf{x}_S)}{\cancel{p_{\theta_0}(\mathbf{x}_B)}\ p_{\theta_0}(\mathbf{x}_S)} \approx \log \frac{p_\theta(\mathbf{x}_\mathbf{S})}{p_{\theta_0}(\mathbf{x}_S)}$$

# 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* .




$\log p_\theta(x_d)$ $\qquad$ $\log p_\theta(x_d) - \log p_{\theta_0}(x_d)$

# Likelihood ratio significantly improves OOD detection on genomics data too

| Method | AUROC |
|---|---|
| Likelihood | 0.630 |
| **Likelihood Ratio** | **0.755** |
| Classifier-based p(y\|x) | 0.622 |
| Classifier-based Entropy | 0.622 |
| Classifier-based ODIN | 0.645 |
| Classifier Ensemble 5 | 0.673 |
| Classifier-based Mahalanobis Distance | 0.496 |

- 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]

[1] Christopher M Bishop. Novelty Detection and Neural Network Validation. 1994.

[2] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. In *NeurIPS*, 2018.

[3] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017.

[4] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, and Balaji Lakshminarayanan. Detecting out-of-distribution inputs to deep generative models using a test for typicality. *arXiv preprint arXiv:1906.02994*, 2019.

[5] Eric Nalisnick, Akihiro Matsukawa, YeeWhye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do Deep Generative Models Know What They Don't Know? In *ICLR*, 2019.

[6] Eric Nalisnick, Akihiro Matsukawa, YeeWhye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Hybrid models with deep and invertible features. In *ICML*, 2019.

[7] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *NeurIPS*, 2019.

[8] Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark A DePristo, Joshua V Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *NeurIPS*, 2019.

[9] Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixel CNN decoders. In *NeurIPS*, 2016.