

# Do Deep Generative Models Know What They Don't Know?

Balaji Lakshminarayanan

[balajiln@](mailto:balajiln@deepmind.com)

Joint work with Eric Nalisnick, Akihiro Matsukawa, Yee  
Whye Teh, Dilan Gorur



# Quantifying Uncertainty In Deep Learning

- Why predictive uncertainty?
  - Good uncertainty scores quantify when we can trust the model's predictions

# Quantifying Uncertainty In Deep Learning

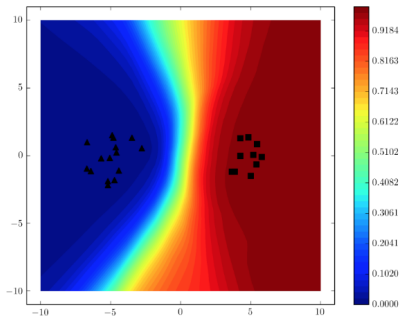
- Why predictive uncertainty?
  - Good uncertainty scores quantify when we can trust the model's predictions
- Predict output distribution  $p(y|x)$  rather than point estimate
  - Classification: output label  $y^*$  along with confidence
  - Regression: output mean and variance

## Source of uncertainty: Inherent stochasticity

Output  $y$  for a given  $x$  could be inherently stochastic

- Rewards in a casino
- Measurement noise in  $y$
- Noise in labeling process (outcome could depend on rater)
- Also known as **aleatoric uncertainty**
- Considered to be “irreducible uncertainty”: persists even in the limit of infinite data

# Source of uncertainty: Model uncertainty



- Multiple values of parameters could be consistent with the observed data
- Also known as **epistemic uncertainty**
- Considered to be “reducible uncertainty”: vanishes in the limit of infinite data (subject to identifiability)

# Applications of Predictive Uncertainty<sup>1</sup>

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

---

<sup>1</sup>Weight uncertainty is also useful, e.g. compression, sensitivity analysis

# Applications of Predictive Uncertainty<sup>1</sup>

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Active learning for efficient data collection

---

<sup>1</sup>Weight uncertainty is also useful, e.g. compression, sensitivity analysis

# Applications of Predictive Uncertainty<sup>1</sup>

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

---

<sup>1</sup>Weight uncertainty is also useful, e.g. compression, sensitivity analysis



# Applications of Predictive Uncertainty<sup>1</sup>

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

---

<sup>1</sup>Weight uncertainty is also useful, e.g. compression, sensitivity analysis

# Applications of Predictive Uncertainty<sup>1</sup>

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

---

<sup>1</sup>Weight uncertainty is also useful, e.g. compression, sensitivity analysis

# Applications of Predictive Uncertainty<sup>1</sup>

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Active learning for efficient data collection
- Dealing with noisy data and train-test skew in production systems
- Reinforcement learning: (Safe) Exploration
- Model interpretability and visualization
- Build modular systems that **know what they don't know**

---

<sup>1</sup>Weight uncertainty is also useful, e.g. compression, sensitivity analysis

# Applications of Predictive Uncertainty<sup>1</sup>

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Active learning for efficient data collection
- Dealing with noisy data and train-test skew in production systems
- Reinforcement learning: (Safe) Exploration
- Model interpretability and visualization
- Build modular systems that **know what they don't know**
- ... and many more!

---

<sup>1</sup>Weight uncertainty is also useful, e.g. compression, sensitivity analysis

# How do we measure the quality of uncertainty?

- **Calibration:** Measures how probabilistic forecasts align with observed long-run frequencies
  - Weather forecasting: Of all days where model predicted rain with 80% probability, what fraction did we observe rain?
  - Measures: Calibration curve / Reliability diagrams, Expected calibration error (ECE)

# How do we measure the quality of uncertainty?

- **Calibration:** Measures how probabilistic forecasts align with observed long-run frequencies
  - Weather forecasting: Of all days where model predicted rain with 80% probability, what fraction did we observe rain?
  - Measures: Calibration curve / Reliability diagrams, Expected calibration error (ECE)
- **Robustness to dataset shift:** does the system exhibit higher uncertainty on inputs far away from training data?
  - $p(y|x)$  is typically accurate when  $x \sim p_{train}(x)$ , but can make overconfident errors when asked to predict on **out-of-distribution (OOD)** inputs

# How do we measure the quality of uncertainty?

- **Calibration:** Measures how probabilistic forecasts align with observed long-run frequencies
  - Weather forecasting: Of all days where model predicted rain with 80% probability, what fraction did we observe rain?
  - Measures: Calibration curve / Reliability diagrams, Expected calibration error (ECE)
- **Robustness to dataset shift:** does the system exhibit higher uncertainty on inputs far away from training data?
  - $p(y|x)$  is typically accurate when  $x \sim p_{train}(x)$ , but can make overconfident errors when asked to predict on **out-of-distribution (OOD)** inputs
  - Cross-validation can inflate performance. Need to measure ability of model to reject OOD inputs (e.g. confidence versus accuracy curves).

# How do we measure the quality of uncertainty?

- **Calibration:** Measures how probabilistic forecasts align with observed long-run frequencies
  - Weather forecasting: Of all days where model predicted rain with 80% probability, what fraction did we observe rain?
  - Measures: Calibration curve / Reliability diagrams, Expected calibration error (ECE)
- **Robustness to dataset shift:** does the system exhibit higher uncertainty on inputs far away from training data?
  - $p(y|x)$  is typically accurate when  $x \sim p_{train}(x)$ , but can make overconfident errors when asked to predict on **out-of-distribution (OOD)** inputs
  - Cross-validation can inflate performance. Need to measure ability of model to reject OOD inputs (e.g. confidence versus accuracy curves).



**How do deep networks fare?**

# Deep networks are poorly calibrated

## On Calibration of Modern Neural Networks

Chuan Guo<sup>\*1</sup> Geoff Pleiss<sup>\*1</sup> Yu Sun<sup>\*1</sup> Kilian Q. Weinberger<sup>1</sup>

### Abstract

Confidence calibration – the problem of predicting probability estimates representative of the true correctness likelihood – is important for classification models in many applications. We discover that modern neural networks, unlike those from a decade ago, are poorly calibrated. Through extensive experiments, we observe that depth, width, weight decay, and Batch Normalization are important factors influencing calibration. We evaluate the performance of various post-processing calibration methods on state-of-the-art architectures with image and document classification datasets. Our analysis and experiments not only offer insights into neural network learning, but also provide a simple and straightforward recipe for practical settings: on most datasets, *temperature scaling* – a single-parameter variant of Platt Scaling – is surprisingly effective at calibrating predictions.

## 1. Introduction

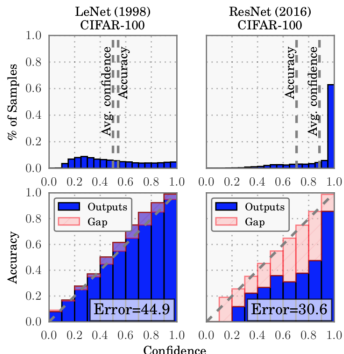


Figure 1. Confidence histograms (top) and reliability diagrams (bottom) for a 5-layer LeNet (left) and a 110-layer ResNet (right) on CIFAR-100. Refer to the text below for detailed illustration.

# High confidence predictions on OOD inputs

## Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images

Anh Nguyen  
University of Wyoming  
anguyen8@uwyo.edu

Jason Yosinski  
Cornell University  
yosinski@cs.cornell.edu

Jeff Clune  
University of Wyoming  
jeffclune@uwyo.edu

### Abstract

Deep neural networks (DNNs) have recently been achieving state-of-the-art performance on a variety of pattern-recognition tasks, most notably visual classification problems. Given that DNNs are now able to classify objects in images with near-human-level performance, questions naturally arise as to what differences remain between computer and human vision. A recent study [30] revealed that changing an image (e.g. of a lion) in a way imperceptible to humans can cause a DNN to label the image as something else entirely (e.g. mislabeling a lion a library). Here we show a related result: it is easy to produce images that are completely unrecognizable to humans, but that state-of-the-art DNNs believe to be recognizable objects with 99.99% confidence (e.g. labeling with certainty that white noise static is a lion). Specifically, we take convolutional neural networks trained to perform well on either the ImageNet or MNIST datasets and then find images with evolutionary neural algorithms or gradient ascent that DNNs label with high confidence as belonging to each dataset class. It is possible to produce images totally unrecognizable to human eyes that DNNs believe with near certainty are familiar objects, which we call “fooling images” (more generally, fooling examples). Our results shed light on interesting differences between human vision and current DNNs, and raise questions about the generality of DNN computer vision.

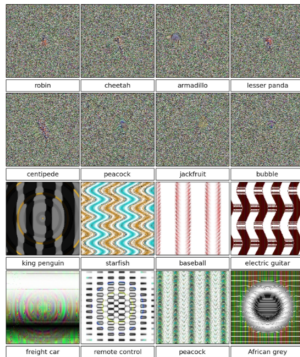


Figure 1. Evolved images that are unrecognizable to humans, but that state-of-the-art DNNs trained on ImageNet believe with  $\geq 99.6\%$  certainty to be a familiar object. This result highlights differences between how DNNs and humans recognize objects. Images are either directly (top) or indirectly (bottom) encoded.

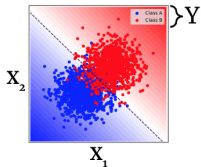
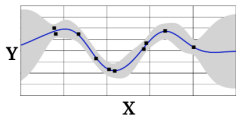
Published as a conference paper at ICLR 2019

---

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

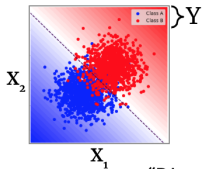
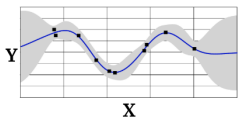
**Eric Nalisnick<sup>‡</sup>, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, Balaji Lakshminarayanan\***  
DeepMind

## So far: Discriminative models



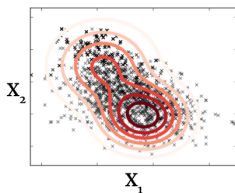
$$p(\mathbf{y}|\mathbf{x})$$

# Discriminative vs Generative models



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

"Discriminative" Model



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

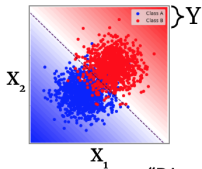
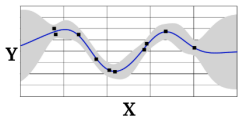
"Generative" Model

- $p(y|x)$  is typically accurate when  $x \sim p_{train}(x)$ , but can make overconfident errors when asked to predict on OOD

---

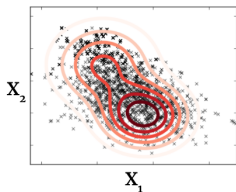
<sup>2</sup>Novelty Detection and Neural Network Validation (Bishop, 1994)

# Discriminative vs Generative models



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

"Discriminative" Model



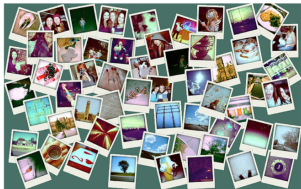
$$p(\mathbf{x})$$

"Generative" Model

- $p(y|x)$  is typically accurate when  $x \sim p_{train}(x)$ , but can make overconfident errors when asked to predict on OOD
- Use generative model to decide when to trust  $p(y|x)$  [1]<sup>2</sup>

<sup>2</sup>Novelty Detection and Neural Network Validation (Bishop, 1994)

Inputs Unlike Training Data



if  $p(\mathbf{x}^*; \phi) < \tau$ ,  
then reject  $\mathbf{x}^*$



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



## AABI workshop, NeurIPS 2017<sup>3</sup>

Panel Discussion, Advances in Approximate Bayesian Inference (AABI) workshop



---

<sup>3</sup><https://www.youtube.com/watch?v=x1UByHT60mQ&feature=youtu.be&t=46m2s>

# AABI workshop, NeurIPS 2017

ZOUBIN: [The Bishop (1994) procedure] should be built into the software.



# AABI workshop, NeurIPS 2017

**ZOUBIN:** [The Bishop (1994) procedure] should be built into the software.

**MODERATOR:** Isn't that hard?



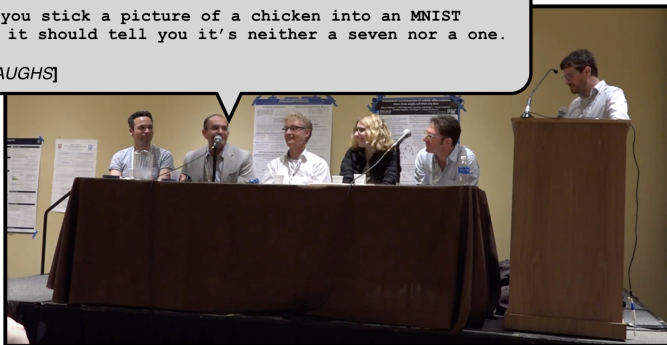
# AABI workshop, NeurIPS 2017

ZOUBIN: [The Bishop (1994) procedure] should be built into the software.

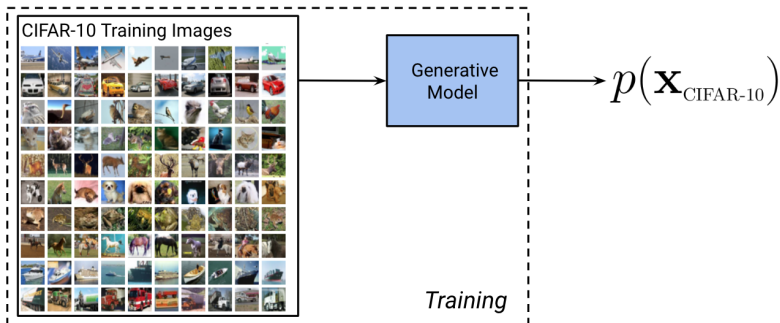
MODERATOR: Isn't that hard?

ZOUBIN: If you stick a picture of a chicken into an MNIST classifier, it should tell you it's neither a seven nor a one.

[AUDIENCE LAUGHS]



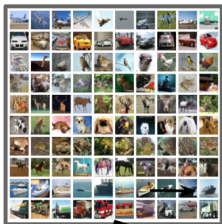
# Generative models for CIFAR



Deep generative models where density  $p(x)$  can be computed:  
**Flows**, Auto-regressive models, VAEs (lower bound)

# Training on CIFAR and Testing on SVHN (OOD)

Training: *CIFAR-10*



Testing: *SVHN*

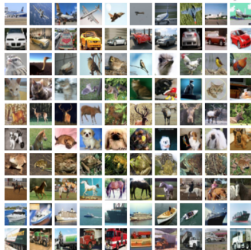


GENERATIVE  
MODEL

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

# Training a Flow-Based Model on CIFAR-10

CIFAR-10 Training Images



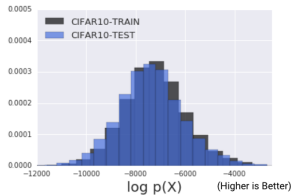
Bits Per Dimension  
( $NLL / \# \text{ dims} / \log 2$ )

---

CIFAR10-Train	3.386
CIFAR10-Test	3.464

---

(Lower is Better)



# Training a Flow-Based Model on CIFAR-10

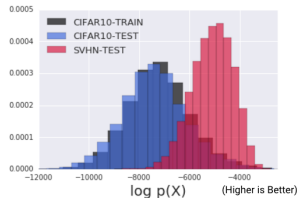
SVHN Test Images



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

CIFAR10-Train	3.386
CIFAR10-Test	3.464
SVHN-Test	<b>2.389</b>

(Lower is Better)





# Training a Flow-Based Model on CIFAR-10

SVHN Test Images



Big Problem!

Bits Per Dimension  
( $NLL / \# \text{ dims} / \log 2$ )

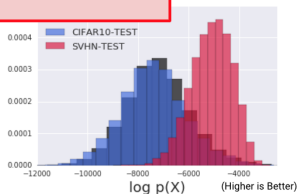
CIFAR10-Train

3.386

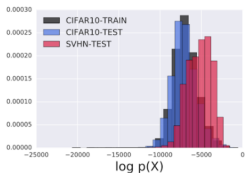
3.464

**2.389**

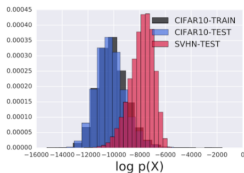
(Lower is Better)



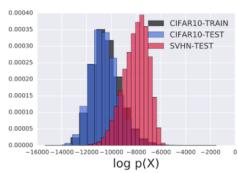
# Phenomenon holds for VAEs and PixelCNN too



(a) PixelCNN

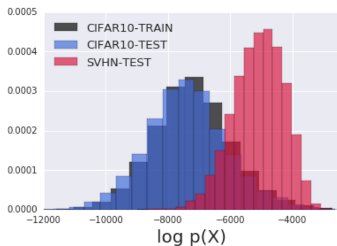


(b) VAE with RNVP as encoder

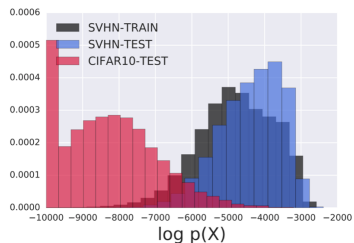


(c) VAE conv-categorical likelihood

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

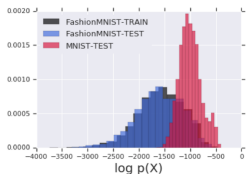


**CIFAR-10 vs SVHN**

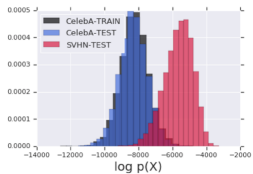


**SVHN vs CIFAR-10**

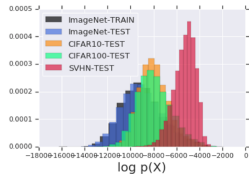
# Additional OOD dataset pairs



**FashionMNIST vs MNIST**

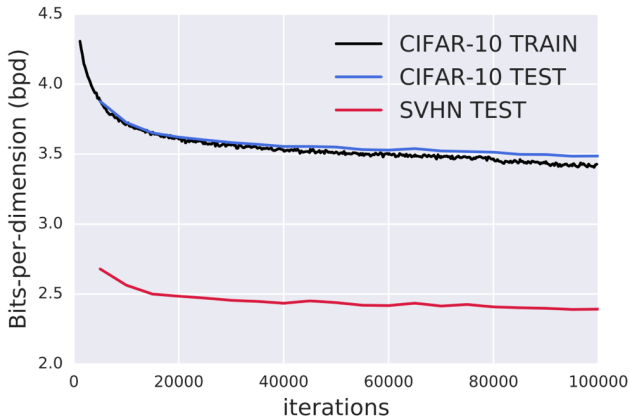


**CelebA vs SVHN**



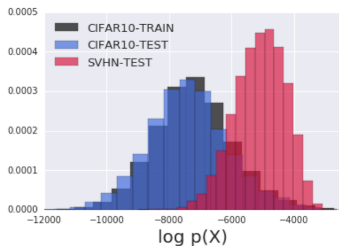
**ImageNet vs CIFAR-10  
vs SVHN**

# Not caused by overfitting: Early stopping does not help

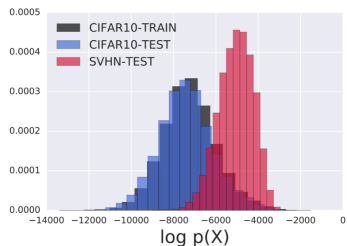


**During Optimization**

# Ensembling does not fix the problem either



**CIFAR-10 vs SVHN**  
*1 Glow*



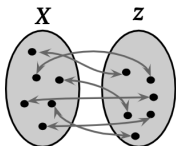
**CIFAR-10 vs SVHN**  
*Ensemble of 10 Glows*

# Digging deeper into flows

# Flows: one slide summary

Define  $Z$  by a transformation of another variable  $X$ :  $Z = f(X)$

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



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

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

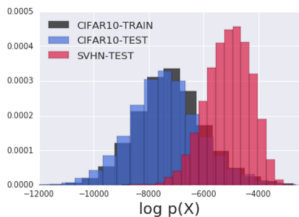
$$p_z(f(X)) \left| \frac{df(X)}{dX} \right| = p(X)$$

Use simple  $p_z$  distribution  
(e.g. standard normal)

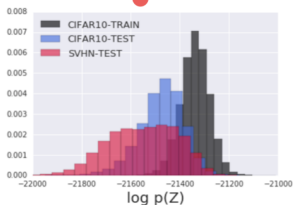
Use  $f$  such that the  
Jacobian  $df/dx$  is easy to  
compute



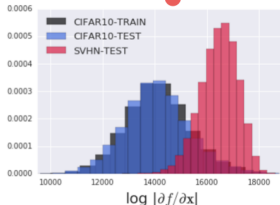
# Decomposition of likelihood for flow models



**CIFAR-10 vs SVHN**

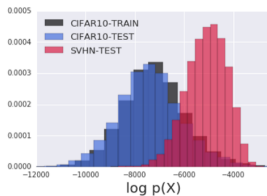


**Distribution Term**



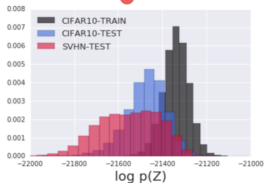
**Volume Term**

# Decomposition of likelihood for flow models

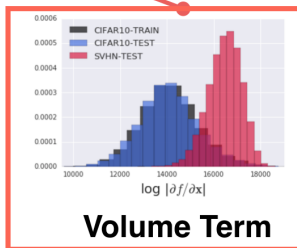


## CIFAR-10 vs SVHN

- Looks to be the cause of the phenomenon



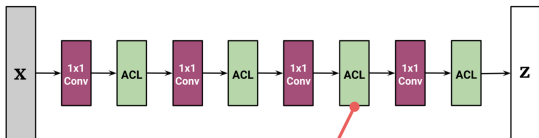
## Distribution Term



## Volume Term

# Is the log volume term the culprit?

We define a sub-class we term *constant-volume* (w.r.t. input) flows.

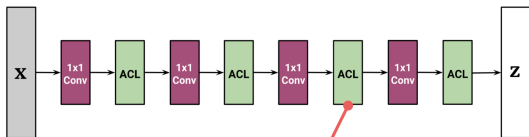


Use only translation operations.

To isolate the effect of the volume term, we define **constant-volume (w.r.t. input) flows**.

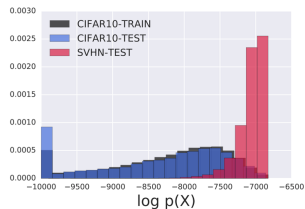
# Is the log volume term the culprit? No.

We define a sub-class we term *constant-volume* (w.r.t. input) flows.



Use only translation operations.

## CIFAR-10 vs SVHN



# **Analysis of Constant Volume GLOW models**

# Analysis of Constant Volume GLOW models

Mathematical characterization:

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

Non-Training  
DistributionTraining  
Distribution

# Analysis of Constant Volume GLOW models

Mathematical characterization:

$$\begin{aligned}
 & 0 < \mathbb{E}_{\underline{\mathbf{q}}} [\log p(\mathbf{x}; \boldsymbol{\theta})] - \mathbb{E}_{\underline{\mathbf{p}^*}} [\log p(\mathbf{x}; \boldsymbol{\theta})] \\
 & \approx \frac{1}{2} \text{Tr} \left\{ \left[ \underbrace{\nabla_{\mathbf{x}_0}^2 \log p_z(f(\mathbf{x}_0; \boldsymbol{\phi})) + \nabla_{\mathbf{x}_0}^2 \log \left| \frac{\partial \mathbf{f}_\phi}{\partial \mathbf{x}_0} \right|}_{\text{Change-of-Variable Terms}} \right] \left( \underbrace{\underline{\boldsymbol{\Sigma}}_{\mathbf{q}} - \overline{\boldsymbol{\Sigma}}_{\mathbf{p}^*}}_{\text{Second Moment of Non-Training Distribution}} \right) \right\}
 \end{aligned}$$

Non-Training Distribution      Training Distribution      Second Moment of Training Distribution

# Analysis of Constant Volume GLOW models

Mathematical characterization:

$$\begin{aligned}
 & 0 < \underbrace{\mathbb{E}_q}_{\text{Non-Training Distribution}}[\log p(\mathbf{x}; \boldsymbol{\theta})] - \underbrace{\mathbb{E}_{p^*}}_{\text{Training Distribution}}[\log p(\mathbf{x}; \boldsymbol{\theta})] \\
 & \approx \frac{1}{2} \text{Tr} \left\{ \underbrace{\left[ \nabla_{\mathbf{x}_0}^2 \log p_z(f(\mathbf{x}_0; \phi)) + \cancel{\nabla_{\mathbf{x}_0}^2 \log p_z(f(\mathbf{x}_0; \phi)) \left[ \frac{\partial f}{\partial \mathbf{x}_0} \right]} \right]}_{\text{Change-of-Variable Terms}} \underbrace{(\underline{\Sigma}_q - \overline{\Sigma}_{p^*})}_{\text{Second Moment of Non-Training Distribution} - \text{Second Moment of Training Distribution}} \right\}
 \end{aligned}$$



# Analysis of Constant Volume GLOW models

Plugging in the CV-Glow transform:

$$\begin{aligned}
 & \text{Tr} \left\{ \left[ \nabla_{\mathbf{x}_0}^2 \log p(\mathbf{x}_0; \boldsymbol{\theta}) \right] (\boldsymbol{\Sigma}_q - \boldsymbol{\Sigma}_{p^*}) \right\} \\
 &= \frac{\partial^2}{\partial \mathbf{z}^2} \log p(\mathbf{z}; \boldsymbol{\psi}) \sum_{c=1}^C \left( \prod_{k=1}^K \sum_{j=1}^C \underbrace{u_{k,c,j}}_{\text{1x1 Conv. Params}} \right)^2 \sum_{h,w} \left( \overline{\sigma_{q,h,w,c}^2} - \overline{\sigma_{p^*,h,w,c}^2} \right)
 \end{aligned}$$

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

# Analysis of Constant Volume GLOW models

Plugging in the CV-Glow transform:

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

$$= \frac{\partial^2}{\partial z^2} \log p(\mathbf{z}; \boldsymbol{\psi}) \sum_{c=1}^C \left( \prod_{k=1}^K \sum_{j=1}^C u_{k,c,j} \right)^2 \sum_{h,w} (\overline{\sigma_{q,h,w,c}^2} - \overline{\sigma_{p^*,h,w,c}^2})$$

Sums over channel dimensions  
 Product over steps in flow  
 1x1 Conv. Params  
 Sum over spatial dimensions  
 Second Moment of Non-Training Distribution  
 Second Moment of Training Distribution

# Analysis of Constant Volume GLOW models

Plugging in the CV-Glow transform:

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

$$= \frac{\partial^2}{\partial z^2} \log p(z; \boldsymbol{\psi}) \sum_{c=1}^C \left( \prod_{k=1}^K \sum_{j=1}^C u_{k,c,j} \right)^2 \sum_{h,w} (\overline{\sigma_{q,h,w,c}^2} - \overline{\sigma_{p^*,h,w,c}^2})$$

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

Non-negative due to square

Second Moment of Non-Training Distribution

Second Moment of Training Distribution

# Analysis of Constant Volume GLOW models

Plugging in the CV-Glow transform:

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

$$= \frac{\partial^2}{\partial z^2} \log p(z; \boldsymbol{\psi}) \sum_{c=1}^C \left( \prod_{k=1}^K \sum_{j=1}^C u_{k,c,j} \right)^2 \sum_{h,w} (\overline{\sigma_{q,h,w,c}^2} - \overline{\sigma_{p^*,h,w,c}^2})$$

**< 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

Sign boils down to difference in moments. Speaks to asymmetric behavior.

# Analysis of Constant Volume GLOW models

Plugging in the CIFAR-10 and SVHN statistics:

$$\mathbb{E}_{\text{SVHN}}[\log p(\mathbf{x}; \boldsymbol{\theta})] - \mathbb{E}_{\text{CIFAR10}}[\log p(\mathbf{x}; \boldsymbol{\theta})]$$
$$\approx \frac{1}{2\sigma_\psi^2} [\alpha_1^2 \cdot 12.3 + \alpha_2^2 \cdot 6.5 + \alpha_3^2 \cdot 14.5] \geq 0 \quad \text{where } \alpha_c = \prod_{k=1}^K \sum_{j=1}^C u_{k,c,j}$$


Differences in variances in the three spatial dimensions

# Analysis of Constant Volume GLOW models

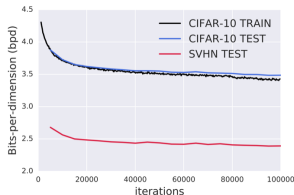
Plugging in the CIFAR-10 and SVHN statistics:

$$\mathbb{E}_{\text{SVHN}}[\log p(\mathbf{x}; \boldsymbol{\theta})] - \mathbb{E}_{\text{CIFAR10}}[\log p(\mathbf{x}; \boldsymbol{\theta})]$$

$$\approx \frac{1}{2\sigma_\psi^2} [\alpha_1^2 \cdot 12.3 + \alpha_2^2 \cdot 6.5 + \alpha_3^2 \cdot 14.5] \geq 0 \quad \text{where } \alpha_c = \prod_{k=1}^K \sum_{j=1}^C u_{k,c,j}$$

Differences in variances in the three spatial dimensions

The expression will be non-negative **for any** parameter setting of the CV flow....



# Analysis of Constant Volume GLOW models

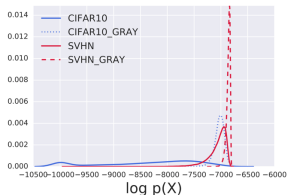
Plugging in the CIFAR-10 and SVHN statistics:

$$\mathbb{E}_{\text{SVHN}}[\log p(\mathbf{x}; \boldsymbol{\theta})] - \mathbb{E}_{\text{CIFAR10}}[\log p(\mathbf{x}; \boldsymbol{\theta})]$$

$$\approx \frac{1}{2\sigma_\psi^2} [\alpha_1^2 \cdot 12.3 + \alpha_2^2 \cdot 6.5 + \alpha_3^2 \cdot 14.5] \geq 0 \quad \text{where } \alpha_c = \prod_{k=1}^K \sum_{j=1}^C u_{k,c,j}$$

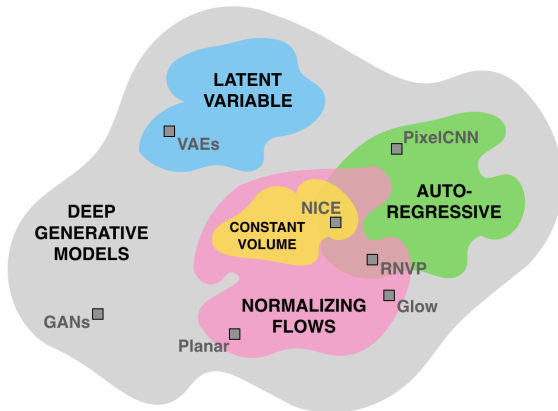
Differences in variances in the three spatial dimensions

This also means that we can manipulate the relative log likelihoods just by changing the variance of the data. For natural images, this amounts to **graying...**



**One weird trick to increase likelihoods: grayscale images!**

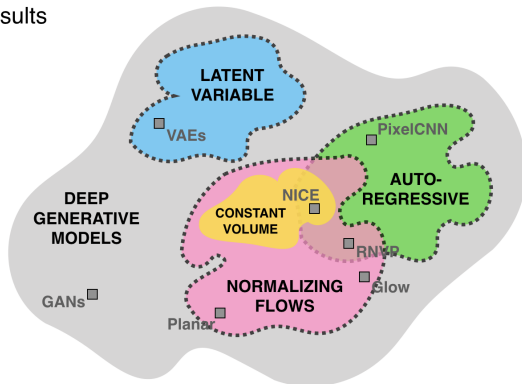
# Summary of Results





# Summary of Results

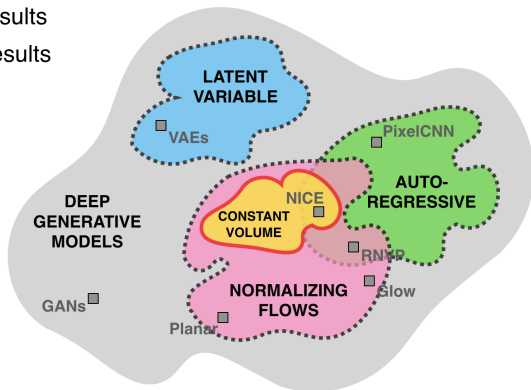
## ■■■ Empirical Results



# Summary of Results

■ ■ ■ Empirical Results

— Analytical Results



## Take home message

- Deep generative models are attractive but have problems detecting out-of-distribution data.

## Take home message

- Deep generative models are attractive but have problems detecting out-of-distribution data.
- For flow-based models, the phenomenon can be explained through the relative variances of the different input distributions
  - Grayscale images
  - Constant images

## Take home message

- Deep generative models are attractive but have problems detecting out-of-distribution data.
- For flow-based models, the phenomenon can be explained through the relative variances of the different input distributions
  - Grayscale images
  - Constant images
- Be cautious when using density estimates from deep generative models as proxy for “similarity” to training data
  - Novelty detection
  - Anomaly detection

## Papers available on my webpage ([link](#))

- *Do Deep Generative Models Know What They Don't Know?*, ICLR, 2019 [[3](#)]
- *Hybrid models with deep and invertible features*, ICML, 2019 [[2](#)]

## Papers available on my webpage ([link](#))

- *Do Deep Generative Models Know What They Don't Know?*, ICLR, 2019 [[3](#)]
- *Hybrid models with deep and invertible features*, ICML, 2019 [[2](#)]

Check out our ICML 2019 workshop

<https://sites.google.com/corp/view/udlworkshop2019/>

# Thanks!

## Acknowledgements:

- Eric Nalisnick
- Akihiro Matsukawa
- Dilan Gorur
- Yee Whye Teh





- [1] Christopher M Bishop. Novelty Detection and Neural Network Validation. 1994.
- [2] Eric Nalisnick, Akihiro Matsukawa, YeeWhye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Hybrid models with deep and invertible features. In *ICML*, 2019.
- [3] Eric Nalisnick, Akihiro Matsukawa, YeeWhye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do Deep Generative Models Know What They Don't Know? In *ICLR*, 2019.