Uncertainty & Out-of-Distribution Robustness in Deep Learning

Balaji Lakshminarayanan

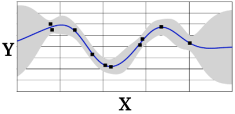
balajiln@

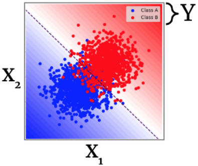
Joint work with colleagues at DeepMind and Google

DeepMind

Part 1: Predictive uncertainty estimation in *Discriminative models*

Discriminative models





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

Quantifying Uncertainty In Deep Learning

- What do we mean by predictive uncertainty? Examples:
  - Classification: output label y^* along with confidence
  - Regression: output mean and variance
- Why predictive uncertainty?
  - **Good uncertainty scores quantify when we can trust the model's predictions**

Sources of predictive uncertainty

- Inherent stochasticity
  - *y* for a given ***x*** could be stochastic, e.g. measurement noise
  - Also known as **aleatoric uncertainty**
  - Considered to be *irreducible uncertainty*: persists even in the limit of infinite data

Sources of predictive uncertainty

- Inherent stochasticity
  - *y* for a given ***x*** could be stochastic, e.g. measurement noise
  - Also known as **aleatoric uncertainty**
  - Considered to be *irreducible uncertainty*: persists even in the limit of infinite data
- 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

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

Applications of Predictive Uncertainty

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Dealing with dataset shift in real-world machine learning systems
  - Feature skew between train and test

Applications of Predictive Uncertainty

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Dealing with dataset shift in real-world machine learning systems
  - Feature skew between train and test
  - *Open-set classification*: May be asked to predict on test inputs that do not belong to any of the training classes

Applications of Predictive Uncertainty

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Dealing with dataset shift in real-world machine learning systems
  - Feature skew between train and test
  - *Open-set classification*: May be asked to predict on test inputs that do not belong to any of the training classes
- Active learning for efficient data collection

Applications of Predictive Uncertainty

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Dealing with dataset shift in real-world machine learning systems
  - Feature skew between train and test
  - *Open-set classification*: May be asked to predict on test inputs that do not belong to any of the training classes
- Active learning for efficient data collection
- Reinforcement learning: (safe) exploration

Applications of Predictive Uncertainty

- Cost-sensitive decision making (e.g. healthcare, self-driving cars, robotics)
- Dealing with dataset shift in real-world machine learning systems
  - Feature skew between train and test
  - *Open-set classification*: May be asked to predict on test inputs that do not belong to any of the training classes
- Active learning for efficient data collection
- Reinforcement learning: (safe) exploration
- Build modular systems that **know what they don't know**

How do we measure the quality of predictive uncertainty?

Challenges

- Lack of ground truth
- Cost of down-stream decisions may be difficult to model

1. Calibration

- Measures how well model's predicted confidence aligns with observed accuracy

1. Calibration

- Measures how well model's predicted confidence aligns with observed accuracy
  - Does *predicted probability of correctness* (confidence) match the *observed frequency of correctness* (accuracy)?

1. Calibration

- Measures how well model's predicted confidence aligns with observed accuracy
  - Does *predicted probability of correctness* (confidence) match the *observed frequency of correctness* (accuracy)?
  - Weather forecasting: Of all days where model predicted rain with 80% probability, what fraction did we observe rain?

1. Calibration

- Measures how well model's predicted confidence aligns with observed accuracy
  - Does *predicted probability of correctness* (confidence) match the *observed frequency of correctness* (accuracy)?
  - Weather forecasting: Of all days where model predicted rain with 80% probability, what fraction did we observe rain?
    - 80% implies perfect calibration

1. Calibration

- Measures how well model's predicted confidence aligns with observed accuracy
  - Does *predicted probability of correctness* (confidence) match the *observed frequency of correctness* (accuracy)?
  - Weather forecasting: Of all days where model predicted rain with 80% probability, what fraction did we observe rain?
    - 80% implies perfect calibration
    - Less than 80% implies model is overconfident

1. Calibration

- Measures how well model's predicted confidence aligns with observed accuracy
  - Does *predicted probability of correctness* (confidence) match the *observed frequency of correctness* (accuracy)?
  - Weather forecasting: Of all days where model predicted rain with 80% probability, what fraction did we observe rain?
    - 80% implies perfect calibration
    - Less than 80% implies model is overconfident
    - Greater than 80% implies model is under-confident

1. Calibration

- Measures how well model's predicted confidence aligns with observed accuracy
  - Does *predicted probability of correctness* (confidence) match the *observed frequency of correctness* (accuracy)?
  - Weather forecasting: Of all days where model predicted rain with 80% probability, what fraction did we observe rain?
    - 80% implies perfect calibration
    - Less than 80% implies model is overconfident
    - Greater than 80% implies model is under-confident
  - Calibration curve / Reliability diagrams
  - Expected calibration error (ECE)

2. Robustness to dataset shift

- Does the system exhibit higher uncertainty on inputs far away from training data?
  - We expect $p(y|\boldsymbol{x})$ to be more accurate when $x \sim p_{TRAIN}(x)$, than on **out-of-distribution (OOD)** inputs

2. Robustness to dataset shift

- Does the system exhibit higher uncertainty on inputs far away from training data?
  - We expect $p(y|\boldsymbol{x})$ to be more accurate when $x \sim p_{TRAIN}(x)$, than on **out-of-distribution (OOD)** inputs
  - Need to measure ability of model to reject OOD inputs.

How do deep networks fare?

Deep networks are poorly calibrated

On Calibration of Modern Neural Networks

Chuan Guo [*1] **Geoff Pleiss** [*1] **Yu Sun** [*1] **Kilian Q. Weinberger** [1]

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.

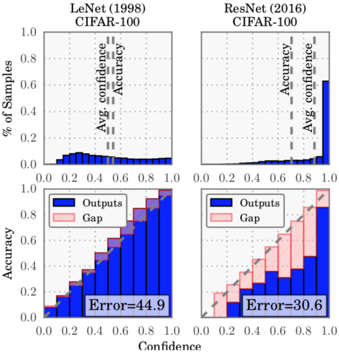


Figure 1. Confidence histograms (top) and reliability diagrams (bottom) for a 5-layer LeNet (left) and a 110-layer ResNet (right) on CIFAR-100. Refer to the text below for detailed illustration.

1. Introduction

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 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.

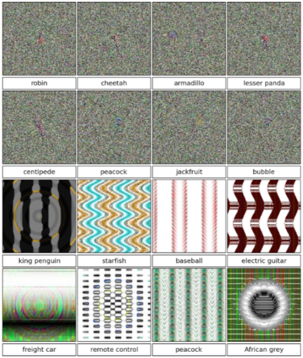


Figure 1. Evolved images that are unrecognizable to humans, but that state-of-the-art DNNs trained on ImageNet believe with $\geq$ 99.6% certainty to be a familiar object. This result highlights differences between how DNNs and humans recognize objects. Images are either directly (*top*) or indirectly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

Predictive Uncertainty in Deep Learning: Large-Scale Benchmark

Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift

Yaniv Ovadia[*]
Google Research
yovadia@google.com

Emily Fertig[*†]
Google Research
emilyaf@google.com

Jie Ren[†]
Google Research
jjren@google.com

Zachary Nado
Google Research
znado@google.com

D Sculley
Google Research
dsculley@google.com

Sebastian Nowozin
Google Research
nowozin@google.com

Joshua V. Dillon
Google Research
jvdillon@google.com

Balaji Lakshminarayanan[‡]
DeepMind
balajiln@google.com

Jasper Snoek[‡]
Google Research
jsnoek@google.com

Popular methods

- (*Vanilla*) Maximum softmax probability [Hendrycks and Gimpel, 2016]

Popular methods

- (*Vanilla*) Maximum softmax probability [Hendrycks and Gimpel, 2016]
- (*Temp Scaling*) Post-hoc calibration by temperature scaling using *i.i.d*. validation set [Guo et al., 2017, Platt, 1999]

Popular methods

- (*Vanilla*) Maximum softmax probability [Hendrycks and Gimpel, 2016]
- (*Temp Scaling*) Post-hoc calibration by temperature scaling using *i.i.d*. validation set [Guo et al., 2017, Platt, 1999]
- (*Dropout*) Monte-Carlo Dropout [Gal and Ghahramani, 2016, Srivastava et al., 2014] with rate p

Popular methods

- (*Vanilla*) Maximum softmax probability [Hendrycks and Gimpel, 2016]
- (*Temp Scaling*) Post-hoc calibration by temperature scaling using *i.i.d*. validation set [Guo et al., 2017, Platt, 1999]
- (*Dropout*) Monte-Carlo Dropout [Gal and Ghahramani, 2016, Srivastava et al., 2014] with rate p
- (*Deep Ensembles*) Ensembles of M networks trained independently on the entire dataset using random initialization [Lakshminarayanan et al., 2017]

Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles

---

Balaji Lakshminarayanan **Alexander Pritzel** **Charles Blundell**
DeepMind
{balajiln,apritzel,cblundell}@google.com

Popular methods

- (*Vanilla*) Maximum softmax probability [Hendrycks and Gimpel, 2016]
- (*Temp Scaling*) Post-hoc calibration by temperature scaling using *i.i.d*. validation set [Guo et al., 2017, Platt, 1999]
- (*Dropout*) Monte-Carlo Dropout [Gal and Ghahramani, 2016, Srivastava et al., 2014] with rate p
- (*Deep Ensembles*) Ensembles of M networks trained independently on the entire dataset using random initialization [Lakshminarayanan et al., 2017]

Popular methods

- (*Vanilla*) Maximum softmax probability [Hendrycks and Gimpel, 2016]
- (*Temp Scaling*) Post-hoc calibration by temperature scaling using *i.i.d.* validation set [Guo et al., 2017, Platt, 1999]
- (*Dropout*) Monte-Carlo Dropout [Gal and Ghahramani, 2016, Srivastava et al., 2014] with rate p
- (*Deep Ensembles*) Ensembles of M networks trained independently on the entire dataset using random initialization [Lakshminarayanan et al., 2017]
- (*SVI*) Stochastic Variational Bayesian Inference [Blundell et al., 2015, Graves, 2011, Wen et al., 2018].

Popular methods

- (*Vanilla*) Maximum softmax probability [Hendrycks and Gimpel, 2016]
- (*Temp Scaling*) Post-hoc calibration by temperature scaling using *i.i.d.* validation set [Guo et al., 2017, Platt, 1999]
- (*Dropout*) Monte-Carlo Dropout [Gal and Ghahramani, 2016, Srivastava et al., 2014] with rate p
- (*Deep Ensembles*) Ensembles of M networks trained independently on the entire dataset using random initialization [Lakshminarayanan et al., 2017]
- (*SVI*) Stochastic Variational Bayesian Inference [Blundell et al., 2015, Graves, 2011, Wen et al., 2018].
- (LL) Approximate Bayesian inference for the parameters of the last layer only [Riquelme et al., 2018]
  - (*LL SVI*) Mean field SVI on the last layer only
  - (*LL Dropout*) Dropout only on activations before last layer

Datasets and Architectures

- Image classification (*convolutional neural networks*)
  - MNIST
  - CIFAR-10
  - ImageNet

Datasets and Architectures

- Image classification (*convolutional neural networks*)
  - MNIST
  - CIFAR-10
  - ImageNet
- Text classification (*LSTMs*)

Datasets and Architectures

- Image classification (*convolutional neural networks*)
  - MNIST
  - CIFAR-10
  - ImageNet
- Text classification (*LSTMs*)
- Criteo Kaggle Display Ads Challenge (*multi-layer perceptrons*)
  - dataset with class-imbalance

Goals of this benchmark

Questions of interest:

Goals of this benchmark

Questions of interest:

- How trustworthy are the uncertainty estimates of different methods under dataset shift?

Goals of this benchmark

Questions of interest:

- How trustworthy are the uncertainty estimates of different methods under dataset shift?
- Does calibration in the i.i.d. setting translate to calibration under dataset shift?

Goals of this benchmark

Questions of interest:

- How trustworthy are the uncertainty estimates of different methods under dataset shift?
- Does calibration in the i.i.d. setting translate to calibration under dataset shift?
- How do uncertainty and accuracy of different methods vary for different datasets and model architectures?

Goals of this benchmark

Questions of interest:

- How trustworthy are the uncertainty estimates of different methods under dataset shift?
- Does calibration in the i.i.d. setting translate to calibration under dataset shift?
- How do uncertainty and accuracy of different methods vary for different datasets and model architectures?

Release open-source TensorFlow code as well as predictions

- https://github.com/google-research/google-research/tree/master/uq_benchmark_2019

Dataset shift: ImageNet-C

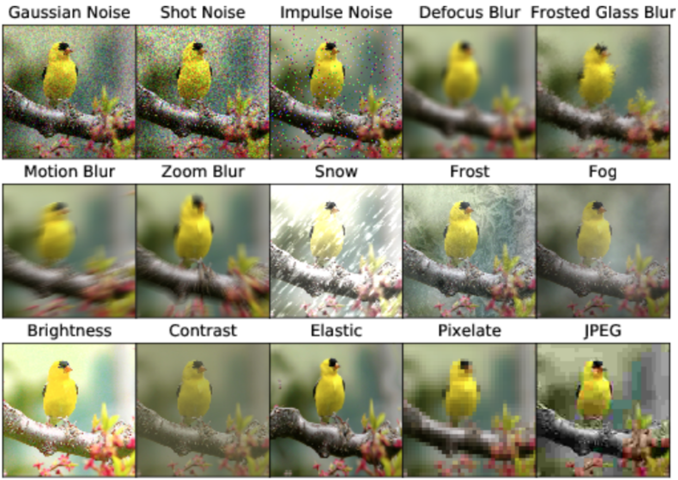


Figure: Image source: [Hendrycks and Dietterich, 2019]

Dataset shift: Varying intensity on ImageNet-C

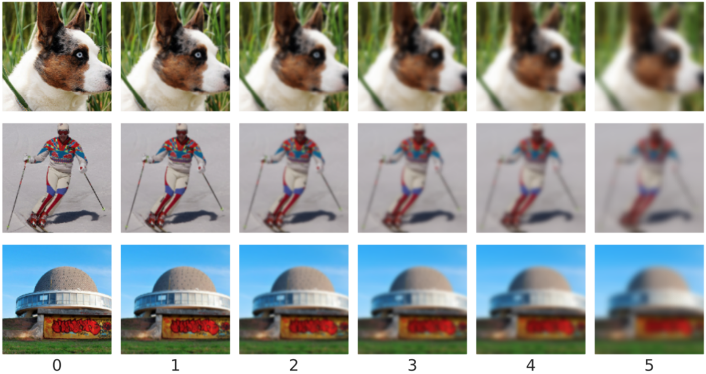
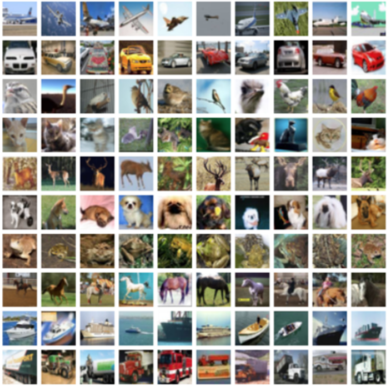


Figure: Increasing intensity of corruption

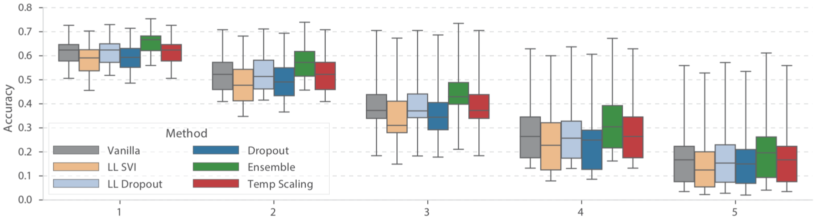
Dataset shift: Testing on completely different dataset

CIFAR-10 Training Images

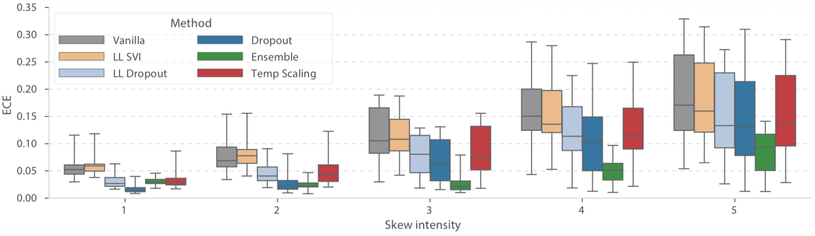


SVHN Test Images

Accuracy decreases as dataset shift increases



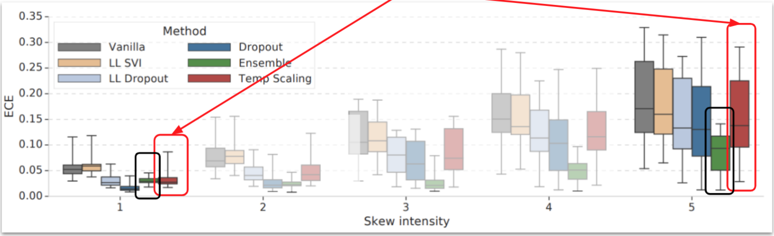
Uncertainty quality decreases significantly as dataset shift increases



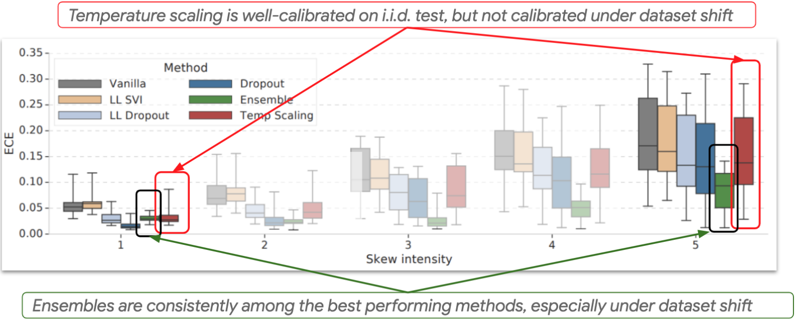
Model is overconfident even though it is way less accurate.

Calibration under dataset shift

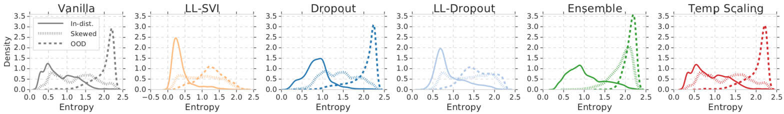
Temperature scaling is well-calibrated on i.i.d. test, but not calibrated under dataset shift

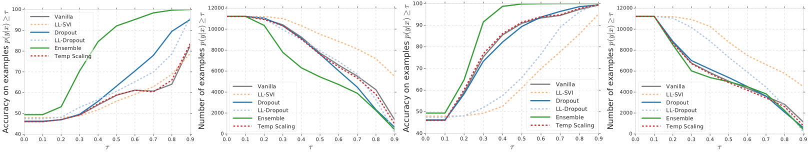


Calibration under dataset shift



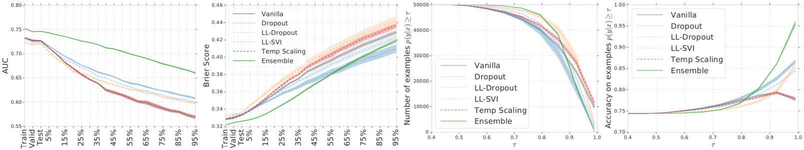
Similar trends on text experiments





(a) Confidence vs Acc. (b) Confidence vs Count (c) Confidence vs Accuracy (d) Confidence vs Count

Similar trends on Criteo experiments as well



Take home messages from our benchmark

- **Calibration under dataset shift** is a major challenge

Take home messages from our benchmark

- **Calibration under dataset shift** is a major challenge
- Relative ordering of methods is mostly consistent (except for MNIST) across our experiments.

Take home messages from our benchmark

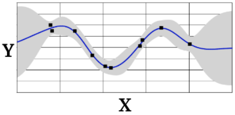
- **Calibration under dataset shift** is a major challenge
- Relative ordering of methods is mostly consistent (except for MNIST) across our experiments.
- Deep ensembles [Lakshminarayanan et al., 2017] seem to perform the best across most metrics and be more robust to dataset shift
  - Relatively small ensemble size (e.g. 5) may be sufficient.

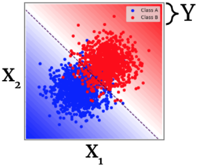
Take home messages from our benchmark

- **Calibration under dataset shift** is a major challenge
- Relative ordering of methods is mostly consistent (except for MNIST) across our experiments.
- Deep ensembles [Lakshminarayanan et al., 2017] seem to perform the best across most metrics and be more robust to dataset shift
  - Relatively small ensemble size (e.g. 5) may be sufficient.
- SVI performs best on MNIST but seems difficult to use on larger datasets (e.g. ImageNet) and architectures (e.g. LSTMs).
  - More work required to make it robust and scalable

Part 2: Out-of-Distribution behavior of *Deep Generative Models*

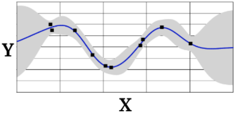
So far: Discriminative models

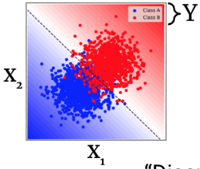




$$p(\mathbf{y}|\mathbf{x})$$

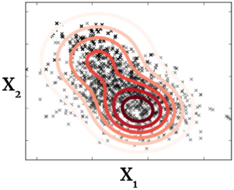
Discriminative vs Generative models





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

“Discriminative” Model

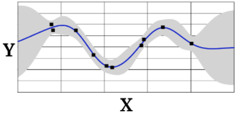


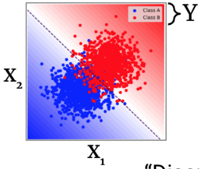
$$p(\mathbf{x})$$

“Generative” Model

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

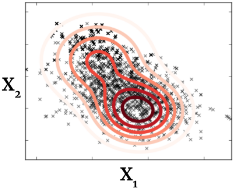
Discriminative vs Generative models





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

"Discriminative" Model

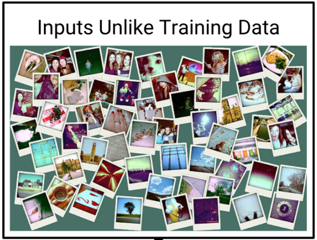


$$p(\mathbf{x})$$

"Generative" Model

- $p(y|\boldsymbol{x})$ is typically accurate when $x \sim p_{TRAIN}(x)$, but can make overconfident errors when asked to predict on OOD
- Use density model $p(\boldsymbol{x})$ to decide when to trust $p(y|\boldsymbol{x})$ [Bishop, 1994]

Novelty Detection & Neural Network Validation



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

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 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 flows to compute exact density $p(\boldsymbol{x})$ and $p(y|\boldsymbol{x})$ in a single feed-forward pass
- Motivation: *If density model $p(\boldsymbol{x})$ can address dataset shift, we can potentially use computationally simpler methods for model uncertainty*

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 flows to compute exact density $p(\boldsymbol{x})$ and $p(y|\boldsymbol{x})$ in a single feed-forward pass
- Motivation: *If density model $p(\boldsymbol{x})$ can address dataset shift, we can potentially use computationally simpler methods for model uncertainty*
- 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]

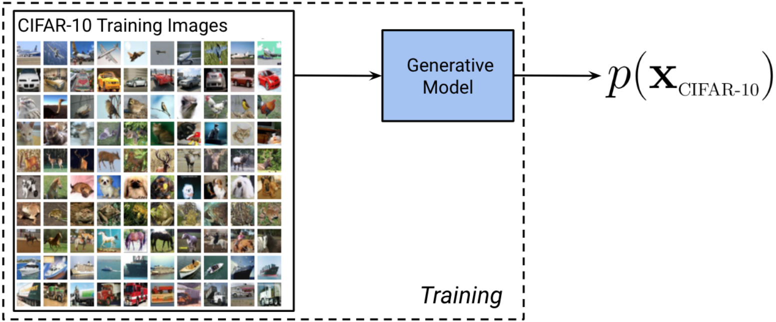
- **Idea**: use flows to compute exact density $p(\boldsymbol{x})$ and $p(y|\boldsymbol{x})$ in a single feed-forward pass
- Motivation: *If density model $p(\boldsymbol{x})$ can address dataset shift, we can potentially use computationally simpler methods for model uncertainty*
- Works well in some 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

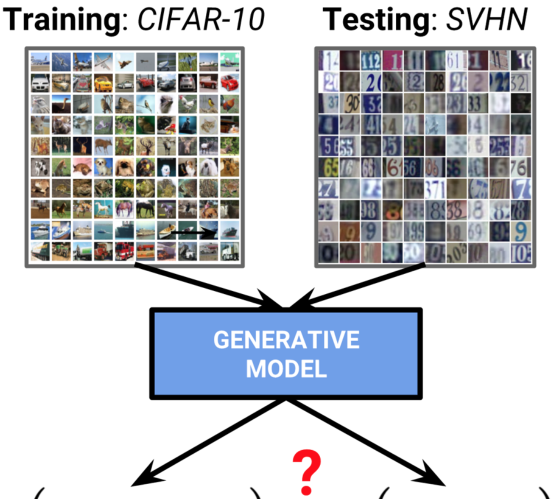
Generative models for CIFAR



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

- Flow-based models: GLOW [Kingma and Dhariwal, 2018]
- Auto-regressive models: PixelCNNs [van den Oord et al., 2016]
- Variational Auto-Encoders (lower bound)

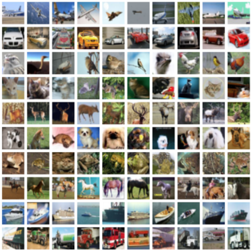
Training on CIFAR and Testing on SVHN (OOD)



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

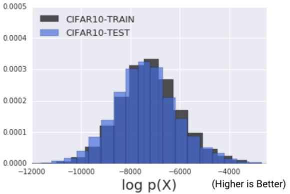
Training a Flow-Based Model on CIFAR-10

CIFAR-10 Training Images



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

(Lower is Better)

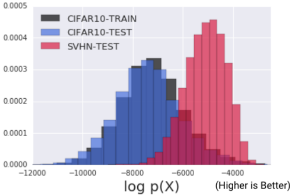


Training a Flow-Based Model on CIFAR-10

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)



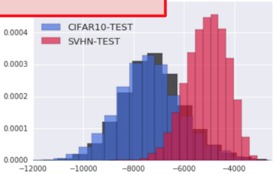
Training a Flow-Based Model on CIFAR-10

SVHN Test Images

	Bits Per Dimension (NLL / # dims / log 2)
CIFAR10-Train	3.386
CIFAR10-T	3.464
	2.389

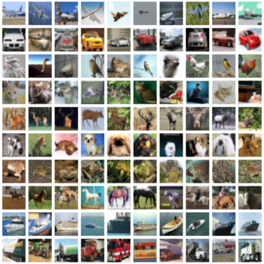
(Lower is Better)

Big Problem!



Model assigns high likelihood to constant inputs too

CIFAR-10 Training Images

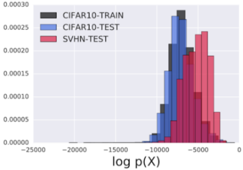


	Bits Per Dimension (NLL / # dims / log 2)
CIFAR10-Train	3.386
CIFAR10-Test	3.464
SVHN-Test	**2.389**

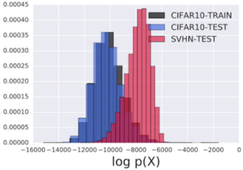
(Lower is Better)

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

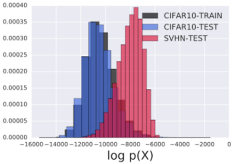
Phenomenon holds for VAEs and PixelCNN too



(a) PixelCNN

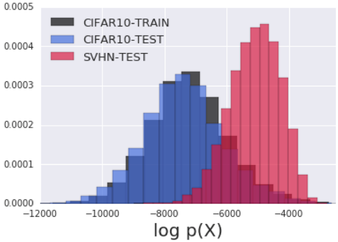


(b) VAE with RNVP as encoder

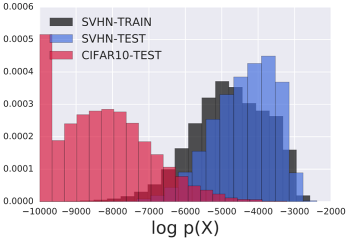


(c) VAE conv-categorical likelihood

The phenomenon is asymmetric w.r.t. datasets

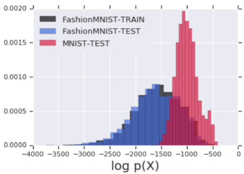


CIFAR-10 vs **SVHN**

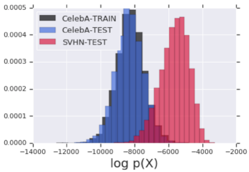


SVHN vs **CIFAR-10**

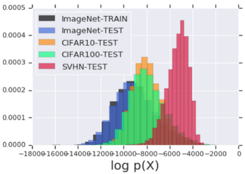
Additional OOD dataset pairs



FashionMNIST vs **MNIST**

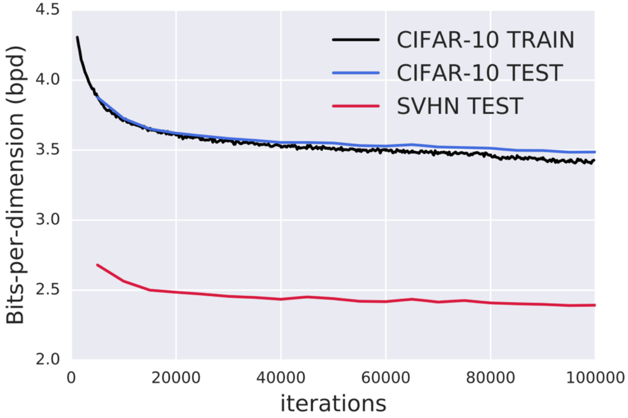


CelebA vs **SVHN**



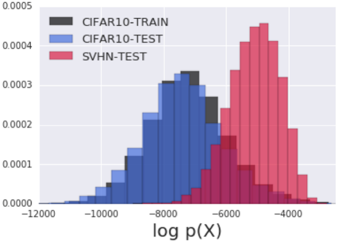
ImageNet vs **CIFAR-10** vs **SVHN**

Phenomenon holds throughout training

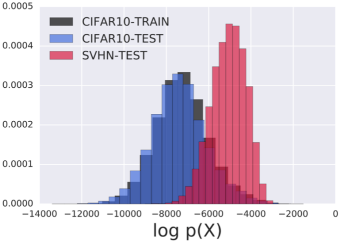


During Optimization

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):

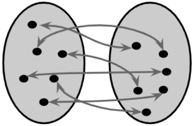
$$p_z(f(X)) \left| \frac{df(X)}{dX} \right| = p(X)$$

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):

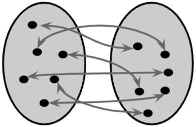
$$p_z(f(X)) \left| \frac{df(X)}{dX} \right| = p(X)$$

Flows: one slide summary

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

$$Z = f(X)$$

f(**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

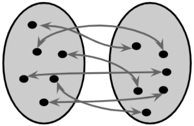
Use architecture such that determinant of Jacobian |df/dx| is easy to compute

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 ($X \rightarrow Z$):

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

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 $f = f_1 \circ f_2 \circ \ldots \circ f_L$

Explaining the observations using CV-GLOW

Mathematical characterization:

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

Explaining the observations using CV-GLOW

Mathematical characterization:

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

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

Explaining the observations using CV-GLOW

Mathematical characterization:

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

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

Explaining the observations using CV-GLOW

Plugging in the CV-Glow transform:

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

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

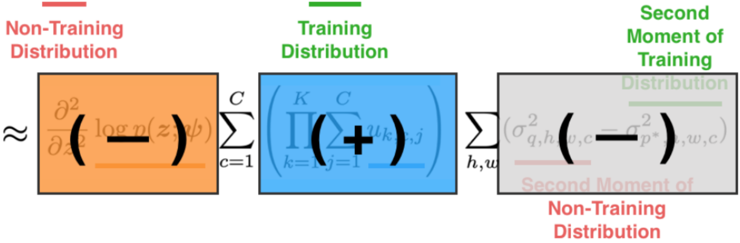
Explaining the observations using CV-GLOW

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

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

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

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

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

- [ ] **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

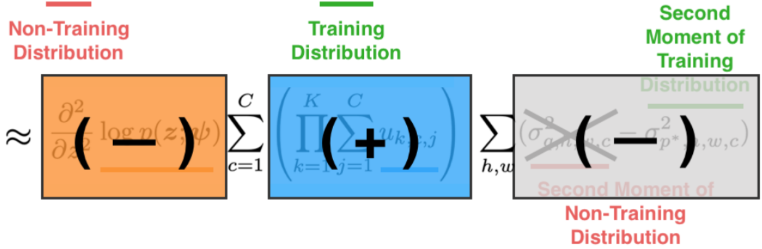
$$0 < \underbrace{\mathbb{E}_q}_{\text{Non-Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \underbrace{\mathbb{E}_{p^*}}_{\text{Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

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

- ☐ **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

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

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

- [ ] **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

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

Second Moment of Training Distribution

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

- [ ] **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

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

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

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

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

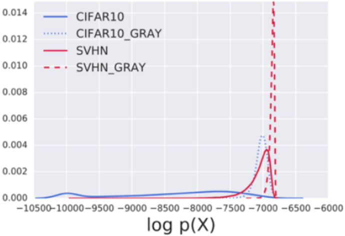
Non-Training Distribution

Training Distribution

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

Second Moment of Training Distribution

Second Moment of Non-Training Distribution



One weird trick to increase your likelihoods!

Take home messages

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

Take home messages

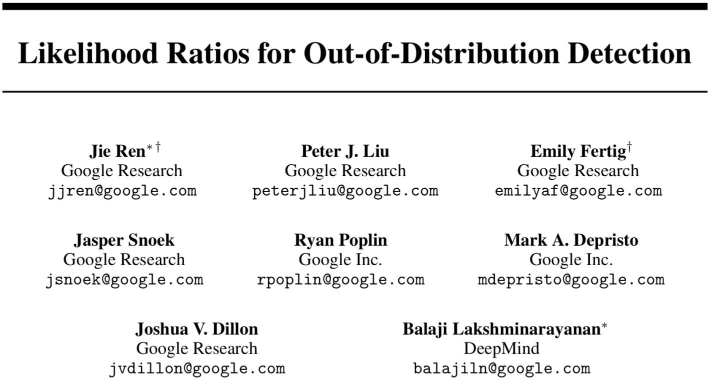
- Deep generative models are attractive but have problems detecting out-of-distribution data.
- Be cautious when using density estimates from deep generative models as proxy for “similarity” to training data
  - Novelty detection
  - Anomaly detection

Take home messages

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

Recent Follow-up Work

Better OOD detection for genomic sequences



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.*

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

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 $\boldsymbol{x}$ 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

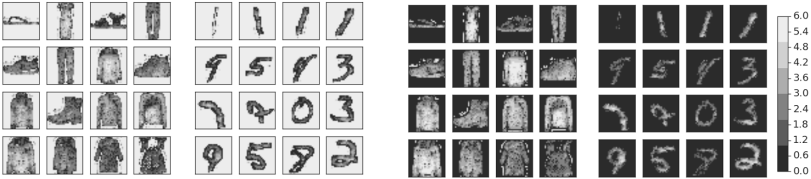
the focus

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

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

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

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

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 μ
  - Typical samples lie near the shell

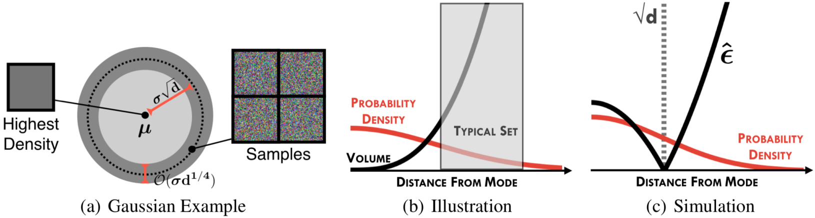
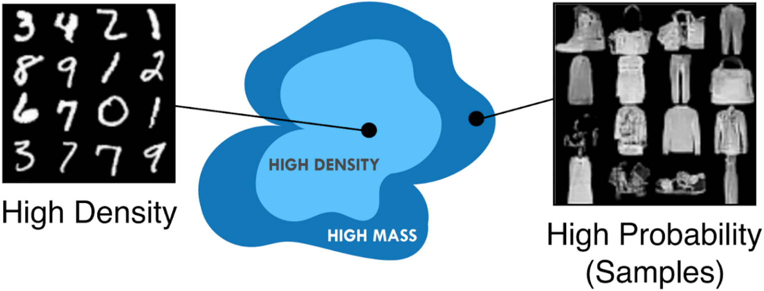


Figure: High dimensional Gaussian

Could similar phenomenon happen with deep generative models too?



Definition of typical sets

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

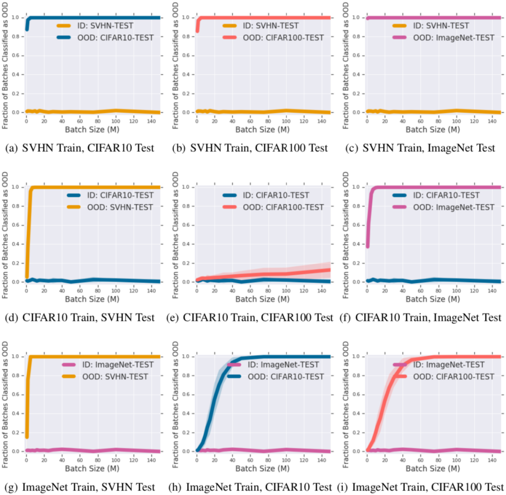
$$\mathbb{H}[p(\mathbf{x})] - \epsilon \leq \frac{-1}{N} \sum_{n=1}^{N} \log p(\boldsymbol{x}_n) \leq \mathbb{H}[p(\mathbf{x})] + \epsilon$$

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



(a) SVHN Train, CIFAR10 Test (b) SVHN Train, CIFAR100 Test (c) SVHN Train, ImageNet Test

(d) CIFAR10 Train, SVHN Test (e) CIFAR10 Train, CIFAR100 Test (f) CIFAR10 Train, ImageNet Test

(g) ImageNet Train, SVHN Test (h) ImageNet Train, CIFAR10 Test (i) ImageNet Train, CIFAR100 Test

Figure: Effect of batch size on AUC of OOD detection

Closing Thoughts

Challenges for Uncertainty Quantification

- Accuracy uncertainty quantification under dataset shift

Challenges for Uncertainty Quantification

- Accuracy uncertainty quantification under dataset shift
- Scalable Bayesian inference

Challenges for Uncertainty Quantification

- Accuracy uncertainty quantification under dataset shift
- Scalable Bayesian inference
- Better understanding out-of-distribution behavior of deep predictive models as well as deep generative models

Challenges for Uncertainty Quantification

- Accuracy uncertainty quantification under dataset shift
- Scalable Bayesian inference
- Better understanding out-of-distribution behavior of deep predictive models as well as deep generative models
- Model mis-specification

Challenges for Uncertainty Quantification

- Accuracy uncertainty quantification under dataset shift
- Scalable Bayesian inference
- Better understanding out-of-distribution behavior of deep predictive models as well as deep generative models
- Model mis-specification
- Realistic benchmarks that reflect real-world challenges

Thanks!

- Aki Matsukawa
- Alex Pritzel
- Charles Blundell
- D. Sculley
- Dilan Gorur
- Emily Fertig
- Eric Nalisnick
- Jasper Snoek
- Jie Ren
- Josh Dillon
- Mark DePristo
- Peter Liu
- Ryan Poplin
- Sebastian Nowozin
- Yaniv Ovadia
- Yee Whye Teh
- Zack Nado

Papers available on my webpage (link)

Predictive uncertainty estimation in deep learning

- *Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift* [Ovadia et al., 2019]
- *Simple and scalable predictive uncertainty estimation using deep ensembles* [Lakshminarayanan et al., 2017]
- *Hybrid models with deep and invertible features* [Nalisnick et al., 2019b]

Papers available on my webpage (link)

Predictive uncertainty estimation in deep learning

- *Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift* [Ovadia et al., 2019]
- *Simple and scalable predictive uncertainty estimation using deep ensembles* [Lakshminarayanan et al., 2017]
- *Hybrid models with deep and invertible features* [Nalisnick et al., 2019b]

Out-of-distribution robustness of deep generative models

- *Do deep generative models know what they don't know?* [Nalisnick et al., 2019a]
- *Likelihood ratios for out-of-distribution detection* [Ren et al., 2019]
- *Detecting out-of-distribution inputs to deep generative models using a test for typicality* [Nalisnick et al., 2019]

Bishop, C. M. (1994). Novelty Detection and Neural Network Validation.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *ICML*.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *ICML*.

Graves, A. (2011). Practical variational inference for neural networks. In *NIPS*.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*.

Hendrycks, D. and Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. *ICLR*.

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

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative Flow with Invertible 1x1 Convolutions. In *NeurIPS*.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*.

Nalisnick, E., Matsukawa, A., Teh, Y., Gorur, D., and Lakshminarayanan, B. (2019a). Do Deep Generative Models Know What They Don't Know? In *ICLR*.

Nalisnick, E., Matsukawa, A., Teh, Y., Gorur, D., and Lakshminarayanan, B. (2019b). Hybrid models with deep and invertible features. In *ICML*.

Nalisnick, E., Matsukawa, A., Teh, Y. W., and Lakshminarayanan, B. (2019). Detecting out-of-distribution inputs to deep generative models using a test for typicality. *arXiv preprint arXiv:1906.02994*.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J. V., Lakshminarayanan, B., and Snoek, J. (2019). Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*.

Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

Ren, J., Liu, P. J., Fertig, E., Snoek, J., Poplin, R., DePristo, M. A., Dillon, J. V., and Lakshminarayanan, B. (2019). Likelihood ratios for out-of-distribution detection. *arXiv preprint arXiv:1906.02845*.

Riquelme, C., Tucker, G., and Snoek, J. (2018). Deep Bayesian Bandits Showdown: An Empirical Comparison of Bayesian Deep Networks for Thompson Sampling. In *ICLR*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *JMLR*.

van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. (2016). Conditional image generation with pixel CNN decoders. In *NeurIPS*.

Wen, Y., Vicol, P., Ba, J., Tran, D., and Grosse, R. (2018). Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*.