Uncertainty & Out-of-Distribution Robustness in Deep Learning

Balaji Lakshminarayanan

balajiln@

Google Brain

Joint work with lots of awesome colleagues at Google and DeepMind

Outline

Motivation and Background

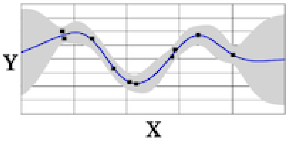
Out-of-Distribution behavior of Deep Generative Models

Uncertainty in Discriminative Models

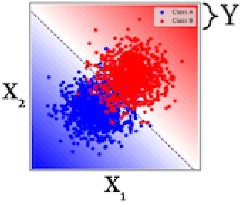
Table of Contents

Quantifying Uncertainty In Deep Learning

- What do we mean by predictive uncertainty? Examples:



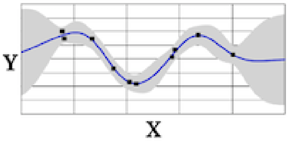
  - Classification: output label y^* along with confidence
  - Regression: output mean and variance



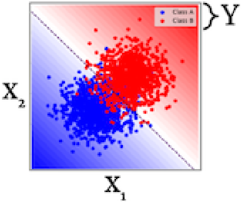
$$p(\mathbf{y}|\mathbf{x})$$

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



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

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

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 example: Of all the days where the 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

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.

1. Introduction

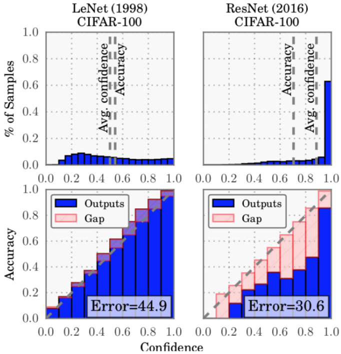


Figure 1. Confidence histograms (top) and reliability diagrams (bottom) for a 5-layer LeNet (left) and a 110-layer ResNet (right) on CIFAR-100. Refer to the text below for detailed illustration.

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 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.

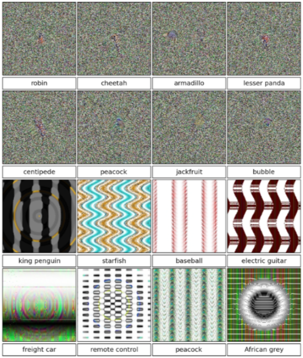


Figure 1. Evolved images that are unrecognizable to humans, but that state-of-the-art DNNs trained on ImageNet believe with $\geq$ 99.6% certainty to be a familiar object. This result highlights differences between how DNNs and humans recognize objects. Images are either directly (*top*) or indirectly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

Applications of neural networks that *know what they don't know*

- Better decision making

Applications of neural networks that *know what they don't know*

- Better decision making
- Dealing with dataset shift in real-world ML systems
  - *Covariate shift* between train and test

Applications of neural networks that *know what they don't know*

- Better decision making
- Dealing with dataset shift in real-world ML systems
  - *Covariate shift* 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 neural networks that *know what they don't know*

- Better decision making
- Dealing with dataset shift in real-world ML systems
  - *Covariate shift* 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
- Improve human-in-the-loop systems

Applications of neural networks that *know what they don't know*

- Better decision making
- Dealing with dataset shift in real-world ML systems
  - *Covariate shift* 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
- Improve human-in-the-loop systems
- Better active learning to collect data in model blindspots

Applications of neural networks that *know what they don't know*

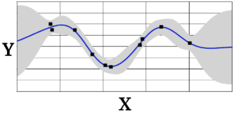
- Better decision making
- Dealing with dataset shift in real-world ML systems
  - *Covariate shift* between train and test
  - *Open-set classification*: May be asked to predict on test inputs that do not belong to any of the training classes
- Improve human-in-the-loop systems
- Better active learning to collect data in model blindspots
- Reinforcement learning: (safe) exploration

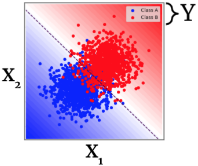
Applications of neural networks that *know what they don't know*

- Better decision making
- Dealing with dataset shift in real-world ML systems
  - *Covariate shift* 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
- Improve human-in-the-loop systems
- Better active learning to collect data in model blindspots
- Reinforcement learning: (safe) exploration
- Build modular systems

Table of Contents

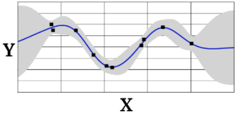
Discriminative models

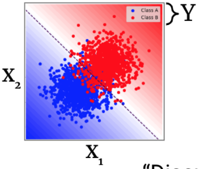




$$p(\mathbf{y}|\mathbf{x})$$

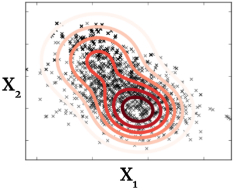
Discriminative vs Generative models





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

"Discriminative" Model

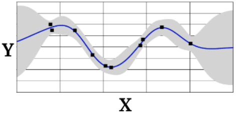


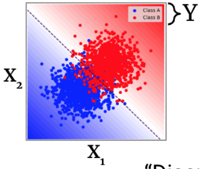
$$p(\mathbf{x})$$

"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

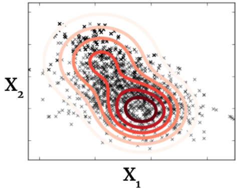
Discriminative vs Generative models





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

"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

Inputs Unlike Training Data

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

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 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
-
  - Density model $p(\boldsymbol{x})$ can address dataset shift,
  - Decompose into two types of 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]

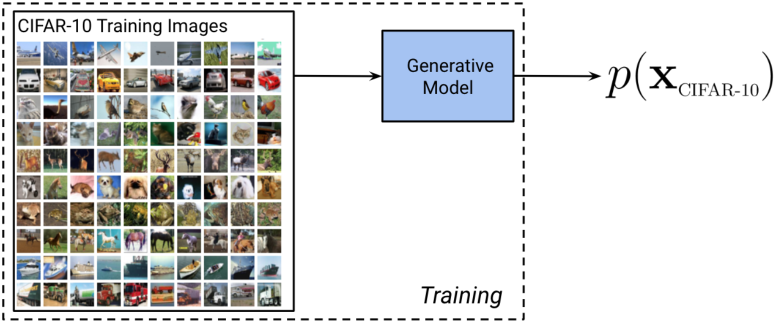
- **Idea**: use normalizing flows to compute exact density $p(\boldsymbol{x})$ and $p(y|\boldsymbol{x})$ in a single feed-forward pass
    - Density model $p(\boldsymbol{x})$ can address dataset shift,
    - Decompose into two types of uncertainty
- Works well in some cases. The failure modes were very 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

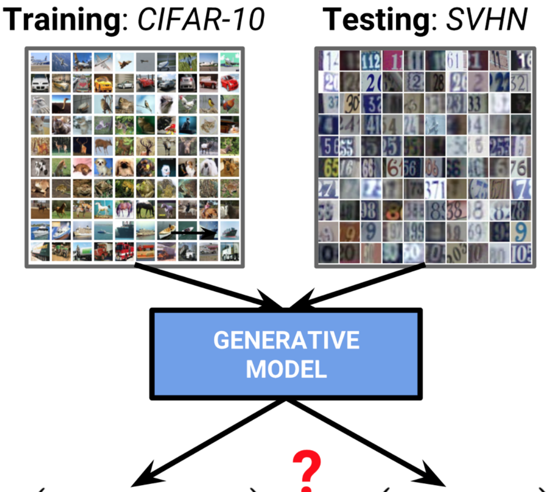
Generative models for CIFAR



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)

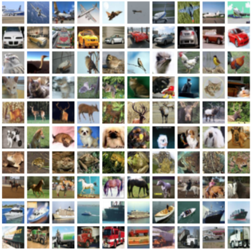
Training on CIFAR and Testing on SVHN (OOD)



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

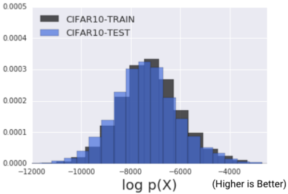
Training a Flow-Based Model on CIFAR-10

CIFAR-10 Training Images



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

(Lower is Better)

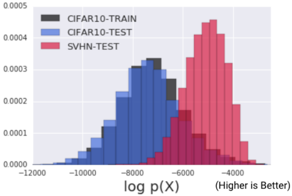


Training a Flow-Based Model on CIFAR-10

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

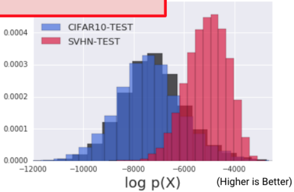
SVHN Test Images

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

CIFAR10-Train	3.386
	3.464
	2.389

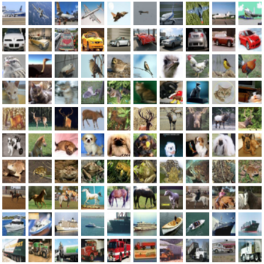
(Lower is Better)

Big Problem!



Model assigns high likelihood to constant inputs too

CIFAR-10 Training Images

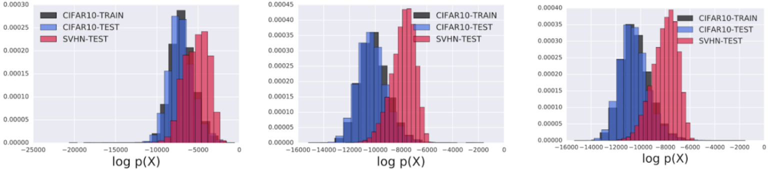


	Bits Per Dimension (NLL / # dims / log 2)
CIFAR10-Train	3.386
CIFAR10-Test	3.464
SVHN-Test	**2.389**

(Lower is Better)

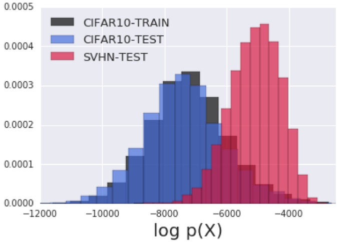
Data Set	Avg. Bits Per Dimension
Glow Trained on CIFAR-10	
Random	15.773
Constant (128)	**0.589**

Phenomenon holds for VAEs and PixelCNN too

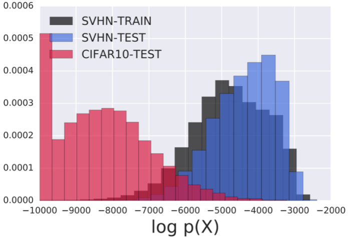


(a) PixelCNN (b) VAE with RNVP as encoder (c) VAE conv-categorical likelihood

The phenomenon is asymmetric w.r.t. datasets

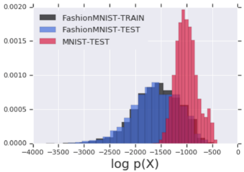


CIFAR-10 vs SVHN

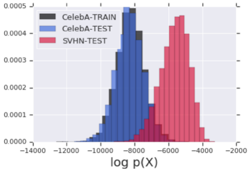


SVHN vs CIFAR-10

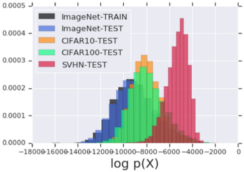
Additional OOD dataset pairs



FashionMNIST vs MNIST

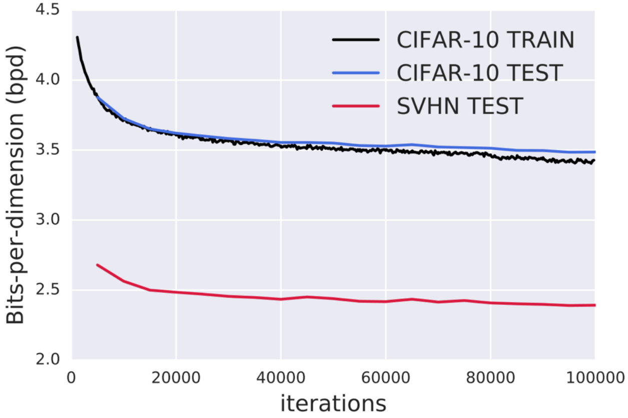


CelebA vs SVHN



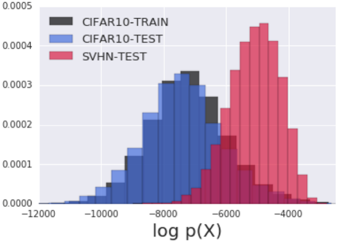
ImageNet vs CIFAR-10 vs SVHN

Phenomenon holds throughout training

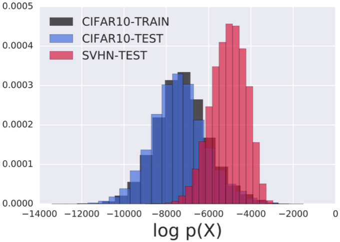


During Optimization

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):

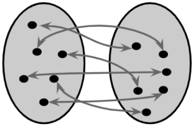
$$p_z(f(X)) \left| \frac{df(X)}{dX} \right| = p(X)$$

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)*



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

Change of Variables Formula (X → Z):

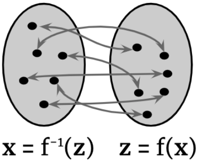
$$p_z(f(X)) \left| \frac{df(X)}{dX} \right| = p(X)$$

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)*



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

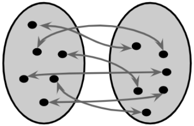
Use architecture such that determinant of Jacobian |df/dx| is easy to compute

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)*



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

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:

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

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

Explaining the observations using Constant Volume GLOW (CV GLOW)

Mathematical characterization:

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

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

Explaining the observations using CV-GLOW

Plugging in the CV-Glow transform:

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

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

Second Moment of Non-Training Distribution

Second Moment of Training Distribution

Explaining the observations using CV-GLOW

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

Non-Training Distribution

Training Distribution

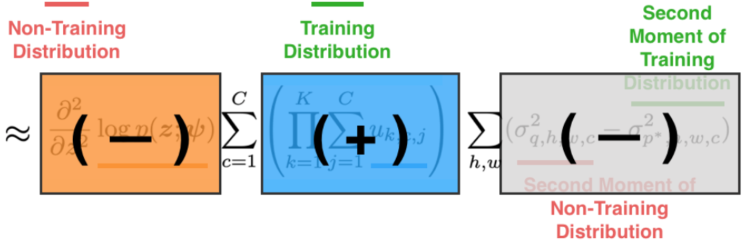
Second Moment of Training Distribution

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

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

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

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

- [ ] **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

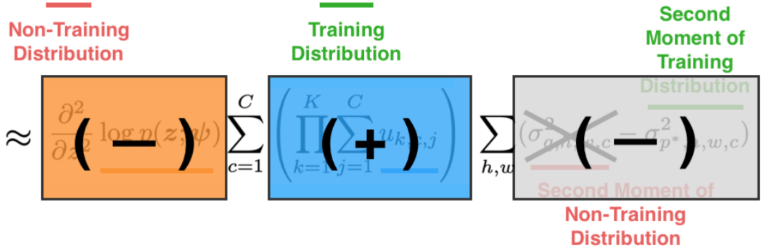
$$0 < \underbrace{\mathbb{E}_q}_{\text{Non-Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \underbrace{\mathbb{E}_{p^*}}_{\text{Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

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

- [ ] **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

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

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

- [ ] **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

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

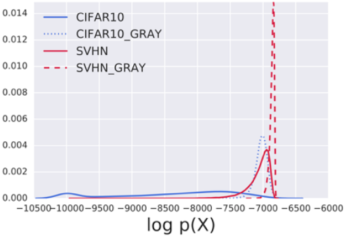
Non-Training Distribution

Training Distribution

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

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

Explaining the failure mode for PixelCNN

Likelihood Ratios for Out-of-Distribution Detection

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

Peter J. Liu
Google Research
peterjliu@google.com

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

Jasper Snoek
Google Research
jsnoek@google.com

Ryan Poplin
Google Inc.
rpoplin@google.com

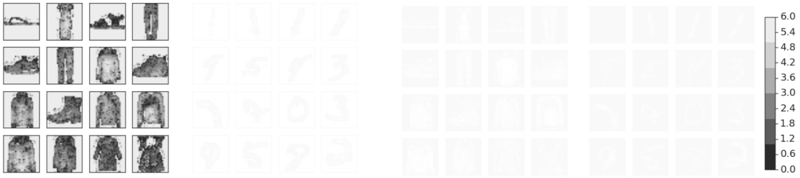
Mark A. Depristo
Google Inc.
mdepristo@google.com

Joshua V. Dillon
Google Research
jvdillon@google.com

Balaji Lakshminarayanan[*]
DeepMind
balajiln@google.com

Explaining the failure mode for PixelCNN

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



$\log p_\theta(x_d)$

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

Explaining the failure mode for PixelCNN

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

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 ***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

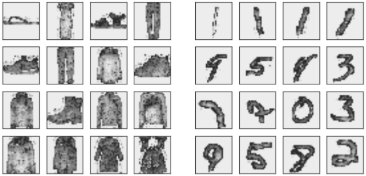
the focus

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

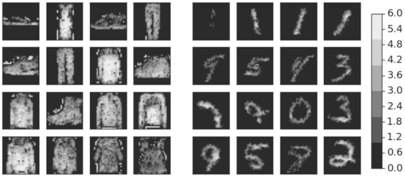
$$\text{LLR}(\mathbf{x}) = \log \frac{p_\theta(\mathbf{x})}{p_{\theta_0}(\mathbf{x})} = \log \frac{\cancel{p_\theta(\mathbf{x}_B)}\ p_\theta(\mathbf{x}_S)}{\cancel{p_{\theta_0}(\mathbf{x}_B)}\ p_{\theta_0}(\mathbf{x}_S)} \approx \log \frac{p_\theta(\mathbf{x}_S)}{p_{\theta_0}(\mathbf{x}_S)}$$

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* .



$\log p_\theta(x_d)$



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

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

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

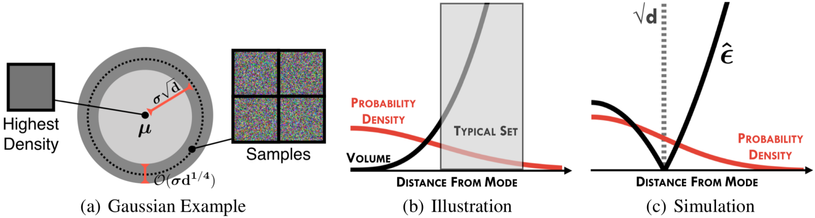
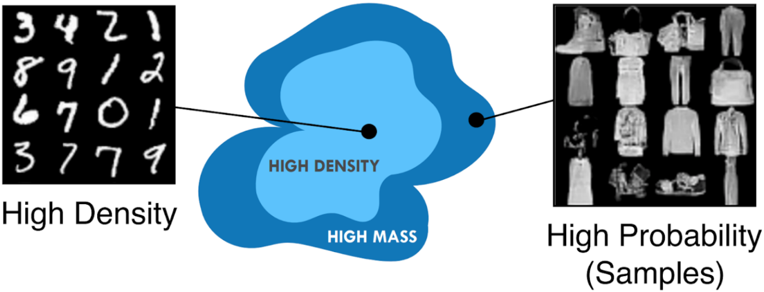


Figure: High dimensional Gaussian

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

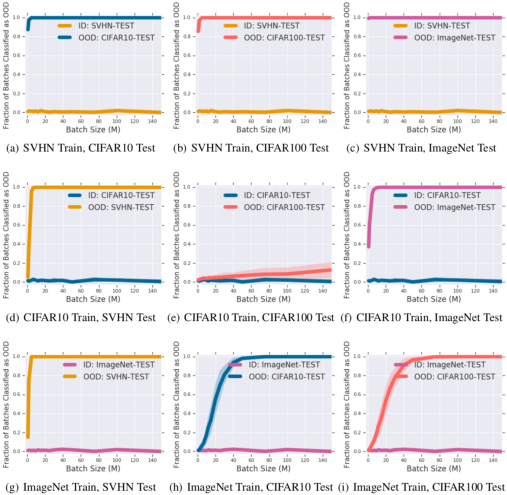
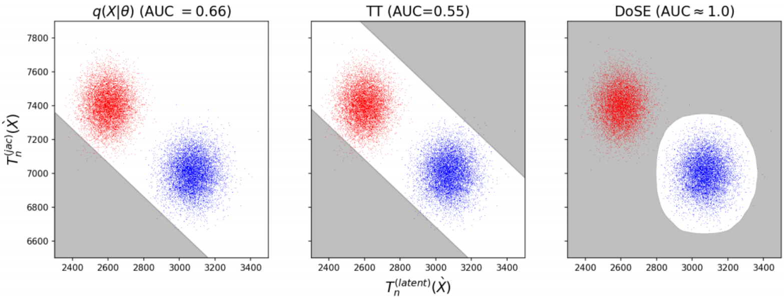


Figure: Effect of batch size on AUC of OOD detection

Density of states for OOD detection



Using multiple statistics can increase power of the test in single-sample setting [Morningstar et al., 2020]

Table of Contents

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]
- (*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
- (*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
- (*Deep Ensembles*) Ensembles of M networks trained independently on the entire dataset using random initialization [Lakshminarayanan et al., 2017]

Datasets and Architectures

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

Goals of this benchmark

Questions of interest:

- How trustworthy are the uncertainty estimates of different methods under dataset shift?
- How do uncertainty and accuracy of different methods vary for different datasets and model architectures?

Dataset shift: ImageNet-C

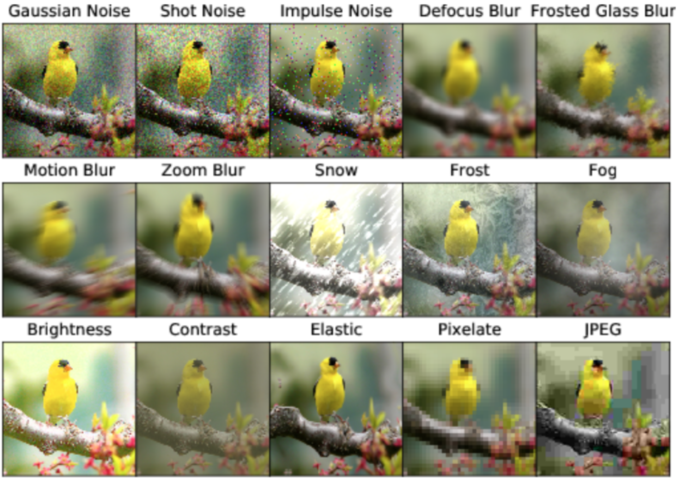


Figure: Image source: [Hendrycks and Dietterich, 2019]

Dataset shift: Varying intensity on ImageNet-C

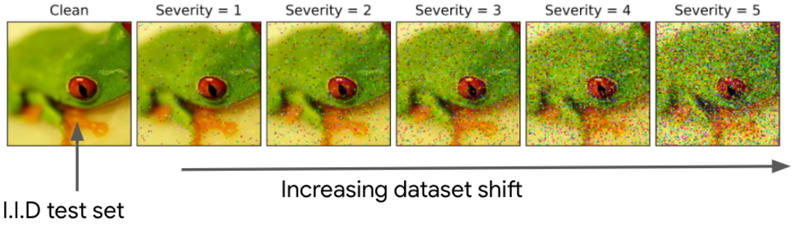
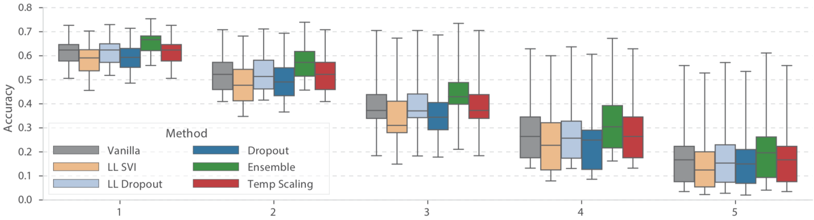
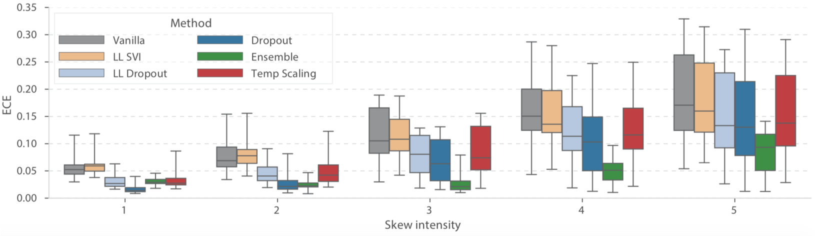


Figure: Increasing intensity of corruption

Accuracy decreases as dataset shift increases



Calibration also decreases significantly as dataset shift increases



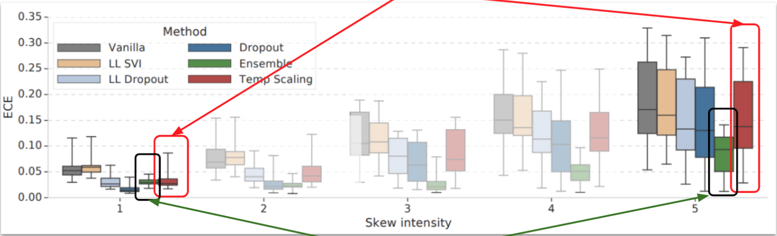
Model is overconfident even though it is way less accurate.

Calibration under dataset shift

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

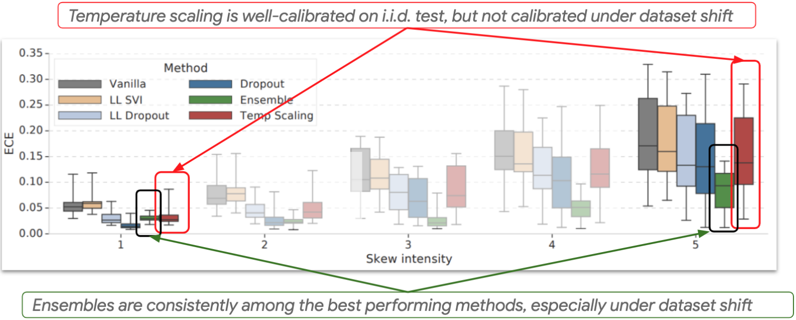
Calibration under dataset shift

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



Ensembles are consistently among the best performing methods, especially under dataset shift

Calibration under dataset shift



We observe similar trends on text and Criteo experiments as well

Results on Criteo experiments

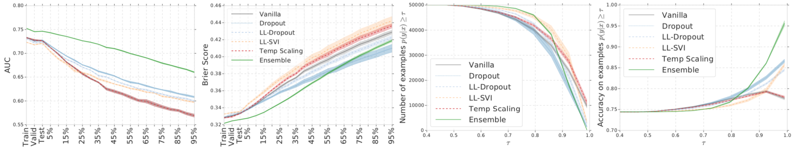
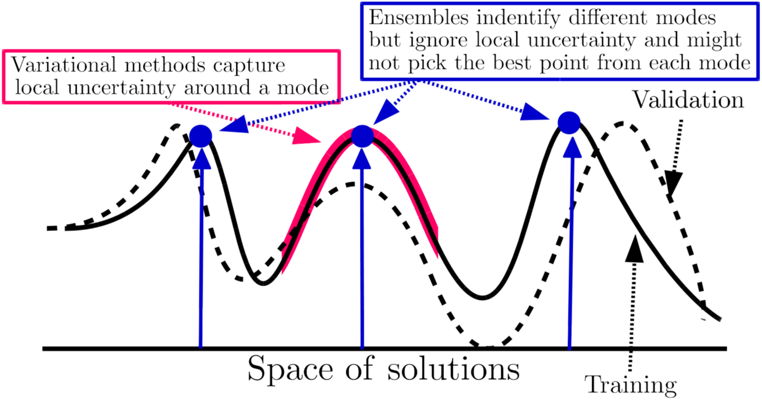
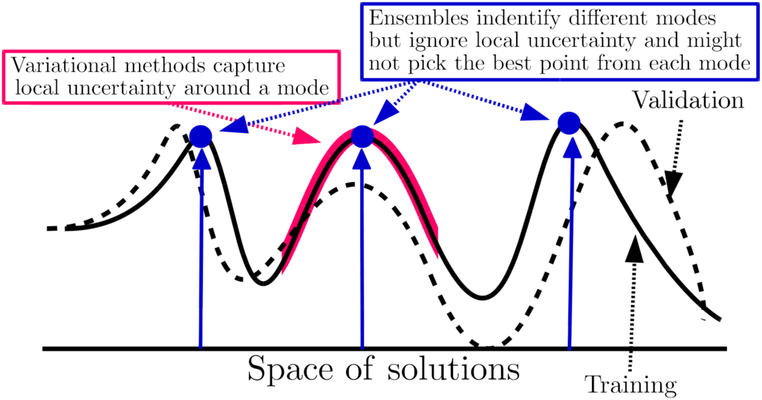


Figure 5: Results on Criteo: The first two plots show degrading AUCs and Brier scores with increasing shift while the latter two depict the distribution of prediction confidences and their corresponding accuracies at 75% randomization of categorical features. SVI is excluded as it performed too poorly.

Deep Ensembles: A Loss landscape perspective [Fort et al., 2019]

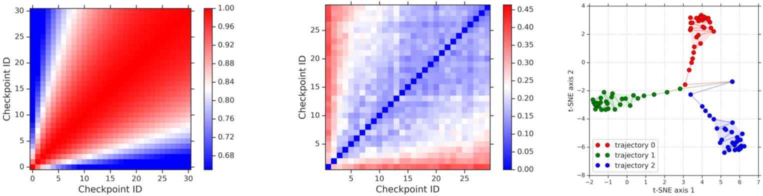


Deep Ensembles: A Loss landscape perspective [Fort et al., 2019]



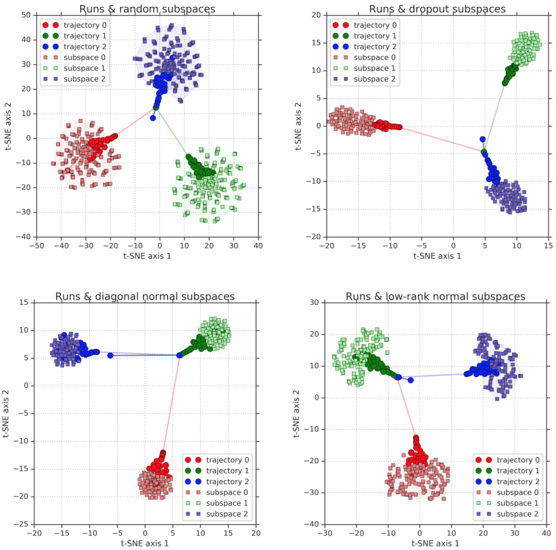
See slides from our talk at the Bayesian deep learning workshop, NeurIPS 2019 for more info.

Function space similarity



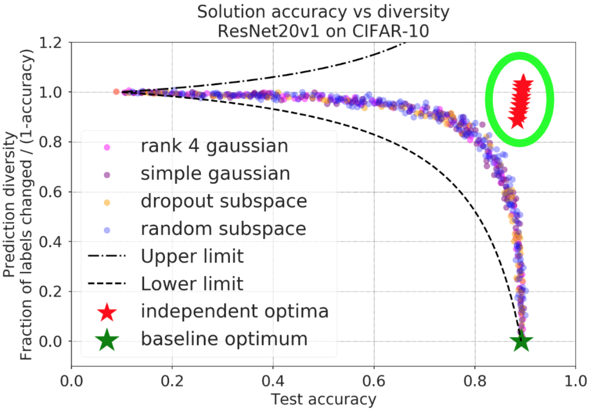
(a) Cosine similarity (weight space) (b) Disagreement (prediction space) (c) t-SNE of predictions

t-SNE of trajectories

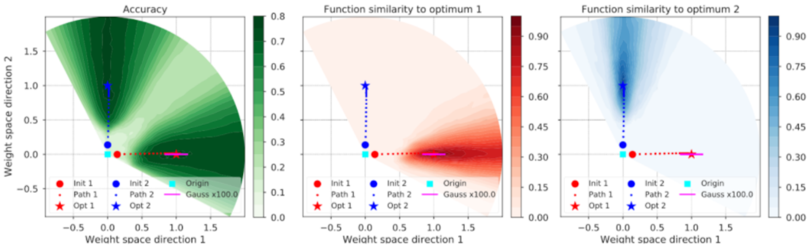


Diversity versus Accuracy trade-off

- Deep ensembles achieve better accuracy versus diversity trade-off than current scalable Bayesian neural nets



Putting it all together

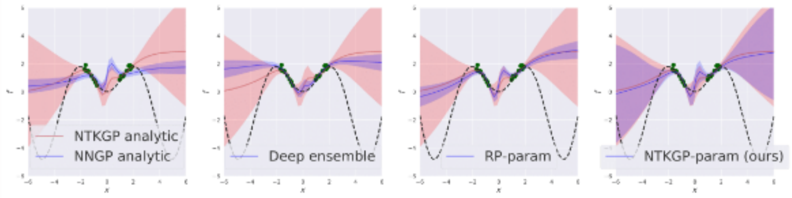


Recent Follow-up Work

Bayesian deep ensembles via the Neural Tangent Kernel

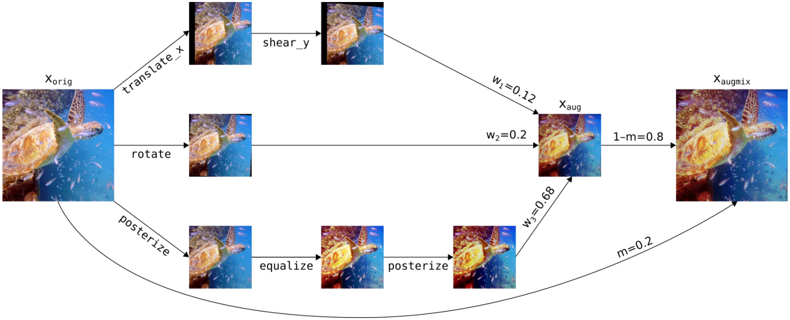
Predictive distributions of wide ensembles for various training methods.

Method	Layers trained	$\mu(\boldsymbol{x})$	$\Sigma(\boldsymbol{x}, \boldsymbol{x}')$
NNGP	Final	$\mathcal{K}_{\boldsymbol{x}\mathcal{X}}(\mathcal{K}_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}\mathcal{Y}$	$\mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'} - \mathcal{K}_{\boldsymbol{x}\mathcal{X}}(\mathcal{K}_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}\mathcal{K}_{\mathcal{X}\boldsymbol{x}'}$
Deep Ensembles	All	$\Theta_{\boldsymbol{x}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}\mathcal{Y}$	$\mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'} - (\Theta_{\boldsymbol{x}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}\mathcal{K}_{\mathcal{X}\boldsymbol{x}'} + h.c.)$ $\Theta_{\boldsymbol{x}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}\mathcal{K}_{\mathcal{X}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}\Theta_{\mathcal{X}\boldsymbol{x}'}$
Randomised Prior	All	$\Theta_{\boldsymbol{x}\mathcal{X}}(\Theta_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}\mathcal{Y}$	$\mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'} - (\Theta_{\boldsymbol{x}\mathcal{X}}(\Theta_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}\mathcal{K}_{\mathcal{X}\boldsymbol{x}'} + h.c.)$ $+\Theta_{\boldsymbol{x}\mathcal{X}}(\Theta_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}(\mathcal{K}_{\mathcal{X}\mathcal{X}}+\sigma^2 I)(\Theta_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}\Theta_{\mathcal{X}\boldsymbol{x}'}$
NTKGP	All (ours)	$\Theta_{\boldsymbol{x}\mathcal{X}}(\Theta_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}\mathcal{Y}$	$\Theta_{\boldsymbol{x}\boldsymbol{x}'} - \Theta_{\boldsymbol{x}\mathcal{X}}(\Theta_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}\Theta_{\mathcal{X}\boldsymbol{x}'}$



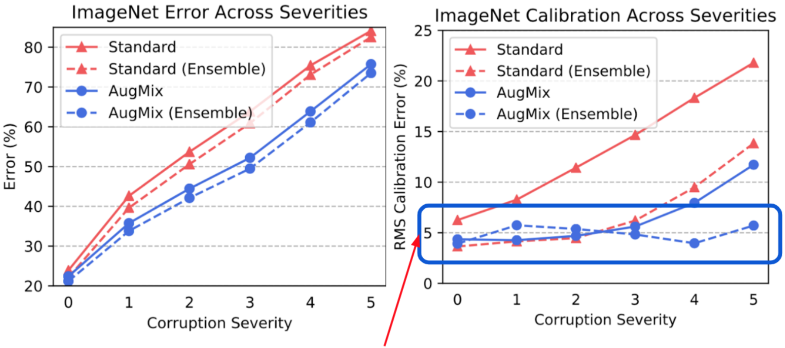
AugMix [Hendrycks et al., 2020]

- Better **data augmentation** (composing base operations and ‘mixing’ them) and **self-supervised learning** can significantly improve calibration under dataset shift.



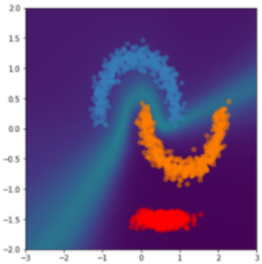
AugMix [Hendrycks et al., 2020]

- Better data augmentation (composing base operations and ‘mixing’ them) and self-supervised learning can **significantly improve calibration under dataset shift**.

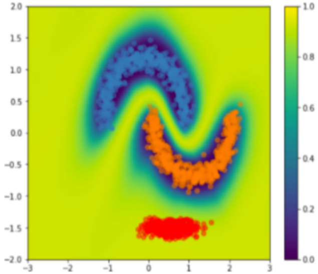


Single model uncertainty [Liu et al., 2020]

- Spectral-normalized Neural Gaussian process for better single model uncertainty $p_\theta(y|x)$
- Replace last-layer with GP layer and add biLipschitz constraint on mapping (Spectral Normalization)



Deep Ensemble



SNGP (single model)

SNGP on text benchmark using BERT

Method	Accuracy (↑)	ECE (↓)	OOD AUROC (↑)	OOD AUPR (↑)	Latency (ms / example)
Deterministic	96.5	0.0236	0.8970	0.7573	**10.42**
MCD-GP	95.9	0.0146	0.9055	0.8030	88.38
DUQ	96.0	0.0585	0.9173	0.8058	15.60
MC Dropout	96.5	0.0210	0.9382	0.7997	85.62
Deep Ensemble	**97.5**	0.0128	0.9635	0.8616	84.46
SNGP	96.6	**0.0115**	**0.9688**	**0.8802**	17.36

SNGP on image benchmark

Method	Accuracy (↑) Clean	Accuracy (↑) Corrupted	ECE (↓) Clean	ECE (↓) Corrupted	NLL (↓) Clean	NLL (↓) Corrupted	OOD AUPR (↑) SVHN	OOD AUPR (↑) CIFAR-100	Latency (↓) (ms / example)
Deterministic	96.0 ± 0.01	72.9 ± 0.01	0.023 ± 0.002	0.153 ± 0.011	0.158 ± 0.01	1.059 ± 0.02	0.781 ± 0.01	0.835 ± 0.01	**3.91**
MC Dropout	96.0 ± 0.01	70.0 ± 0.02	0.021 ± 0.002	0.116 ± 0.009	0.173 ± 0.01	1.152 ± 0.01	0.971 ± 0.01	0.832 ± 0.01	27.10
Deep Ensembles	**96.6 ± 0.01**	**77.9 ± 0.01**	**0.010 ± 0.001**	**0.087 ± 0.004**	**0.114 ± 0.01**	**0.815 ± 0.01**	0.964 ± 0.01	0.888 ± 0.01	38.10
MCD-GP	95.5 ± 0.02	70.0 ± 0.01	0.024 ± 0.004	0.100 ± 0.007	0.172 ± 0.01	1.157 ± 0.01	0.960 ± 0.01	0.863 ± 0.01	29.53
DUQ	94.7 ± 0.02	71.6 ± 0.02	0.034 ± 0.002	0.183 ± 0.011	0.239 ± 0.02	1.348 ± 0.01	0.973 ± 0.01	0.854 ± 0.01	8.68
DNN-SN	96.0 ± 0.01	72.5 ± 0.01	0.025 ± 0.004	0.178 ± 0.013	0.171 ± 0.01	1.306 ± 0.01	0.974 ± 0.01	0.859 ± 0.01	5.20
DNN-GP	95.9 ± 0.01	71.7 ± 0.01	0.029 ± 0.002	0.175 ± 0.008	0.221 ± 0.02	1.380 ± 0.01	0.976 ± 0.01	0.887 ± 0.01	5.58
SNGP (Ours)	95.9 ± 0.01	74.6 ± 0.01	0.018 ± 0.001	0.090± 0.012	0.138 ± 0.01	0.935 ± 0.01	**0.990 ± 0.01**	**0.905 ± 0.01**	6.25

Papers available on my webpage (link)

Predictive uncertainty estimation in deep learning

- *Simple and scalable predictive uncertainty estimation using deep ensembles* [Lakshminarayanan et al., 2017]
- *Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift* [Ovadia et al., 2019]
- *AugMix: A simple data processing method to improve robustness and uncertainty* [Hendrycks et al., 2020]
- *Deep Ensembles: A loss landscape perspective* [Fort et al., 2019]
- *Bayesian Deep Ensembles via the Neural Tangent Kernel* [He et al., 2020]
- *Simple and principled uncertainty estimation with deterministic deep learning via distance awareness* [Liu et al., 2020]

Out-of-distribution robustness of deep generative models

- *Hybrid models with deep and invertible features* [Nalisnick et al., 2019a]
- *Do deep generative models know what they don't know?* [Nalisnick et al., 2019b]
- *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]
- *Density of States Estimation for Out-of-Distribution Detection* [Morningstar et al., 2020]

Extra slides

C. M. Bishop. Novelty Detection and Neural Network Validation. 1994.

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

S. Fort, H. Hu, and B. Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

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

A. Graves. Practical variational inference for neural networks. In *NeurIPS*, 2011.

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

B. He, B. Lakshminarayanan, and Y. W. Teh. Bayesian deep ensembles via the neural tangent kernel. In *NeurIPS*, 2020.

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

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

D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. In *ICLR*, 2020.

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

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

J. Z. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax-Weiss, and B. Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In *NeurIPS*, 2020.

W. R. Morningstar, C. Ham, A. G. Gallagher, B. Lakshminarayanan, A. A. Alemi, and J. V. Dillon. Density of states estimation for out-of-distribution detection. *arXiv preprint arXiv:2006.09273*, 2020.

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

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

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

Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *NeurIPS*, 2019.

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

J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. A. DePristo, J. V. Dillon, and B. Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *NeurIPS*, 2019.

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

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

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

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