Google AI
Brain Team

Practical Uncertainty Estimation & Out-of-Distribution Robustness in Deep Learning

Dustin Tran, Jasper Snoek, Balaji Lakshminarayanan



1. Why uncertainty & robustness?

2. Foundations.

3. Recent advances.

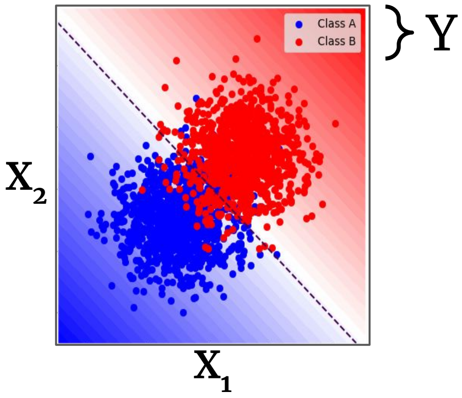
Motivation

What do we mean by Uncertainty?

Return a distribution over predictions rather than a single prediction.

- ***Classification***: Output label along with its confidence.
- ***Regression***: Output mean along with its variance.

Good uncertainty estimates quantify ***when we can trust the model's predictions***.



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

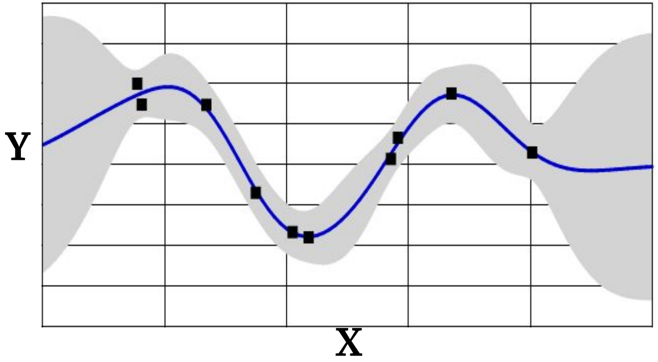


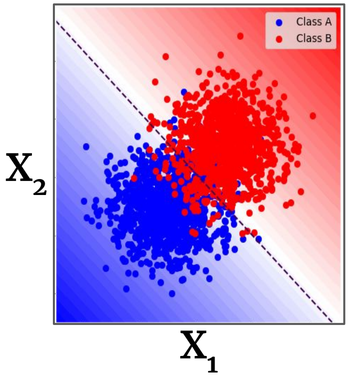
Image credit: Eric Nalisnick

What do we mean by Out-of-Distribution Robustness?

Google AI Brain Team

I.I.D. $p_{TEST}(y,x) = p_{TRAIN}(y,x)$

(*Independent and Identically Distributed*)



O.O.D. $p_{TEST}(y,x) \neq p_{TRAIN}(y,x)$

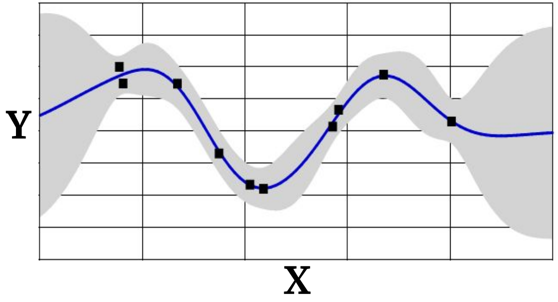


Image credit: Eric Nalisnick

What do we mean by Out-of-Distribution Robustness?

Google AI Brain Team

I.I.D. $p_{TEST}(y,x) = p_{TRAIN}(y,x)$

O.O.D. $p_{TEST}(y,x) \neq p_{TRAIN}(y,x)$

Examples of dataset shift:

- ***Covariate shift.*** Distribution of features p(x) changes and p(y|x) is fixed.
- ***Open-set recognition.*** New classes may appear at test time.
- ***Label shift.*** Distribution of labels p(y) changes and p(x|y) is fixed.

ImageNet-C: Varying Intensity for Dataset Shift

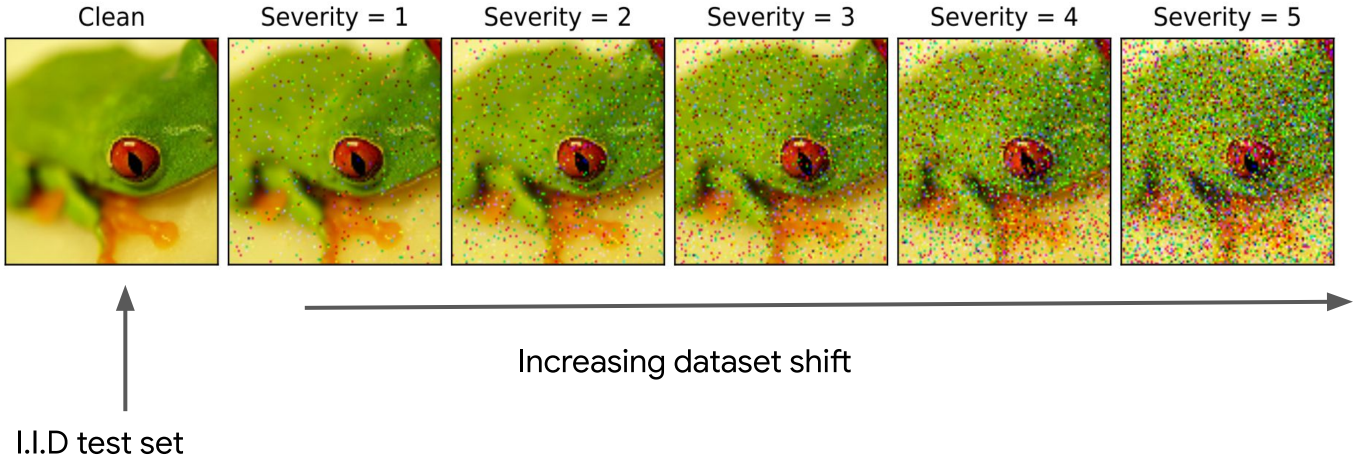
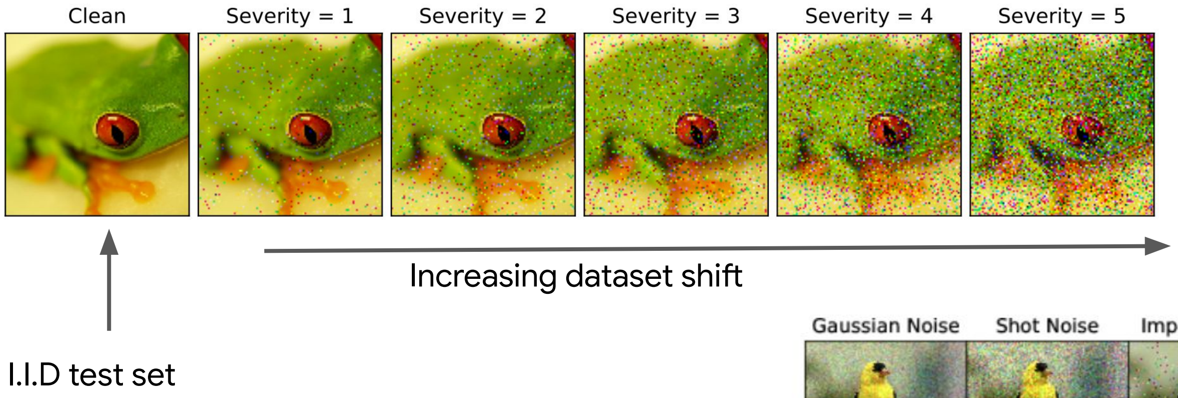


Image source: Benchmarking Neural Network Robustness to Common Corruptions and Perturbations, [Hendrycks & Dietterich, 2019](#).

ImageNet-C: Varying Intensity for Dataset Shift



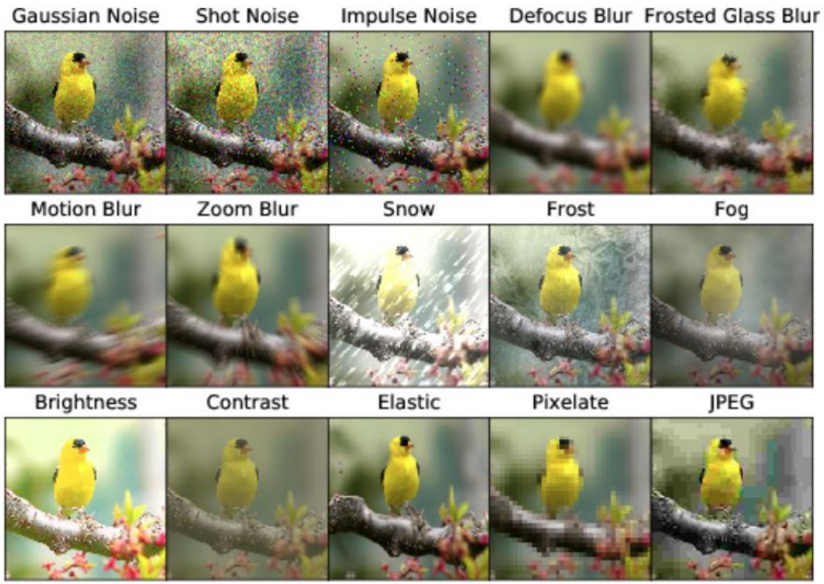
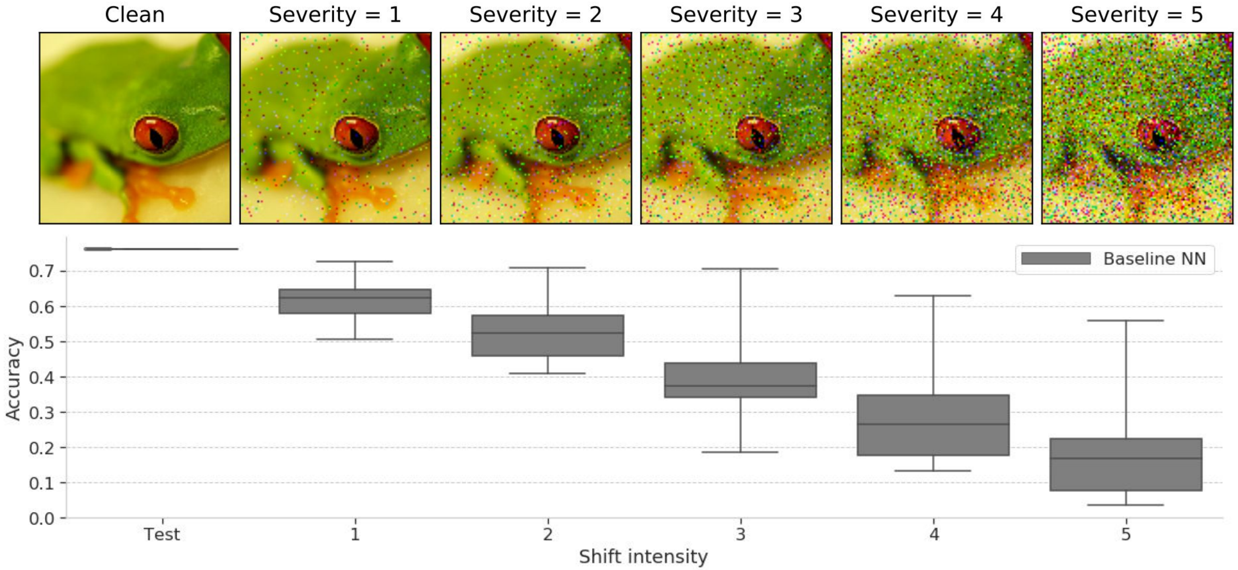


Image source: Benchmarking Neural Network Robustness to Common Corruptions and Perturbations, [Hendrycks & Dietterich, 2019](#).

Neural networks do not generalize under covariate shift

Google AI Brain Team



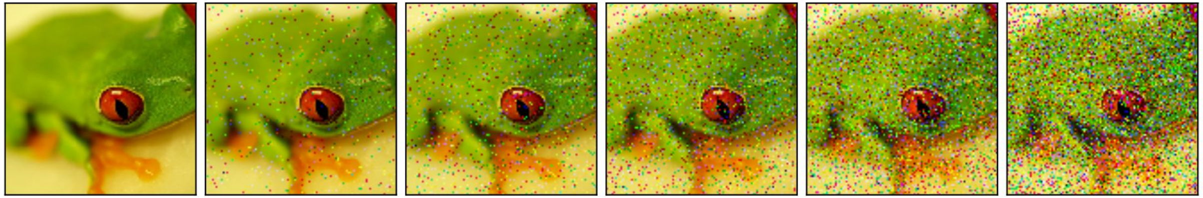
- **Accuracy drops** with increasing shift on Imagenet-C

- But do the models know that they are less accurate?

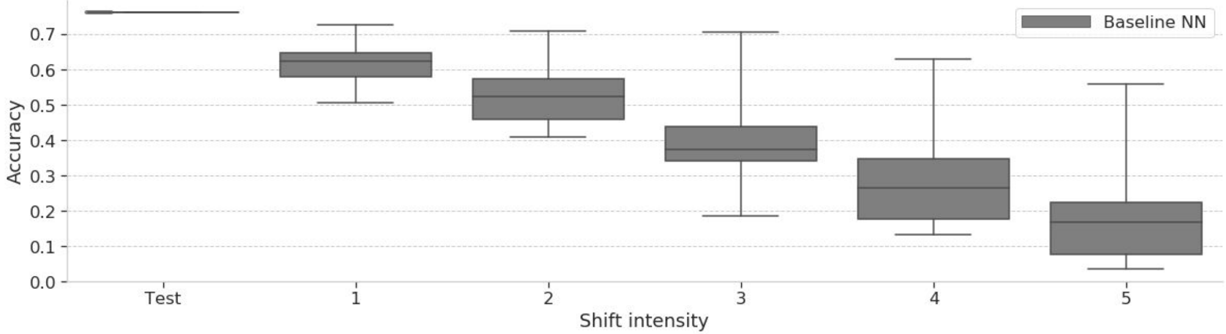
Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift?, [Ovadia et al. 2019]

Neural networks *do not know when they don't know*

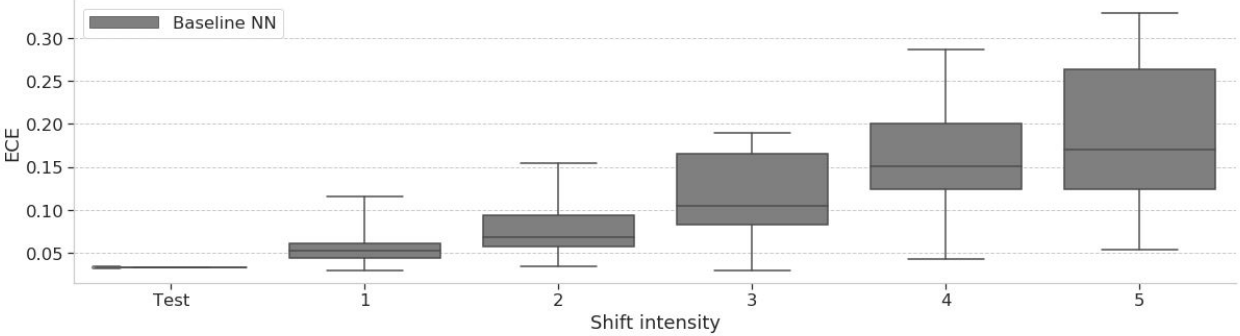
Google AI
Brain Team



- **Accuracy drops** with increasing shift on Imagenet-C



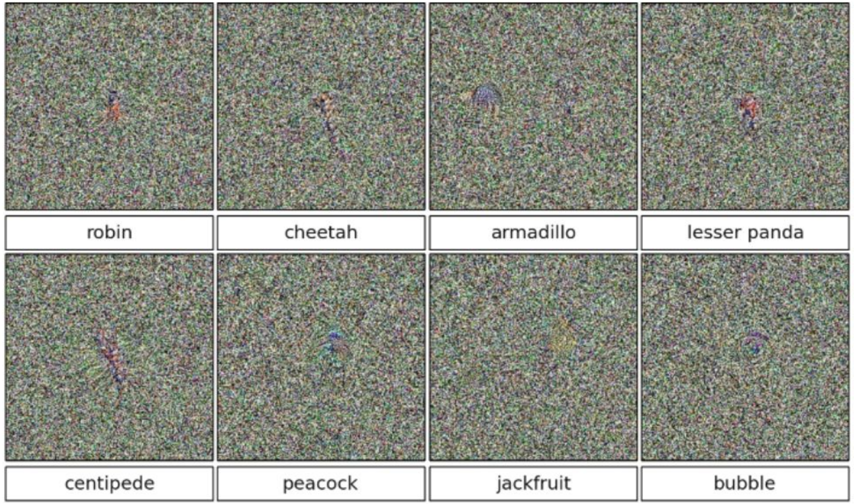
- **Quality of uncertainty degrades with shift** -> "overconfident mistakes"



Models assign high confidence predictions to OOD inputs

Google AI
Brain Team

Example images where model assigns >99.5% confidence.



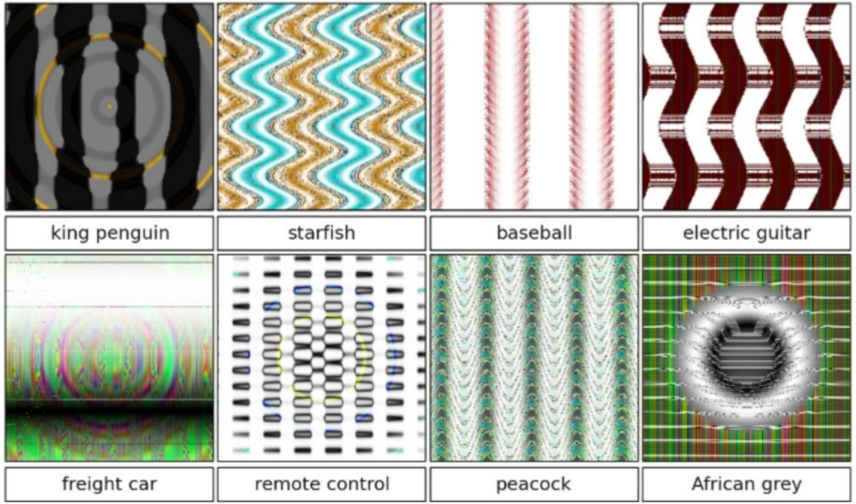
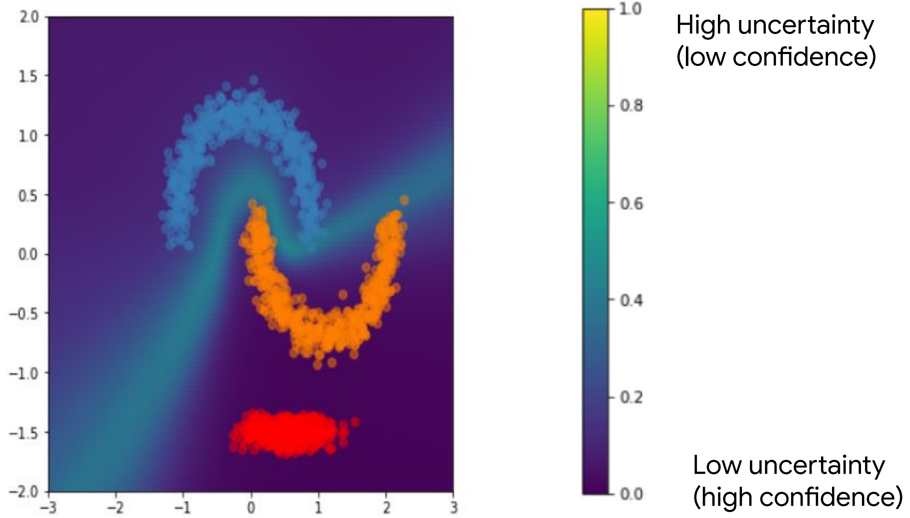


Image source: “Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images” [Nguyen et al. 2014]

Models assign high confidence predictions to OOD inputs

Google AI
Brain Team

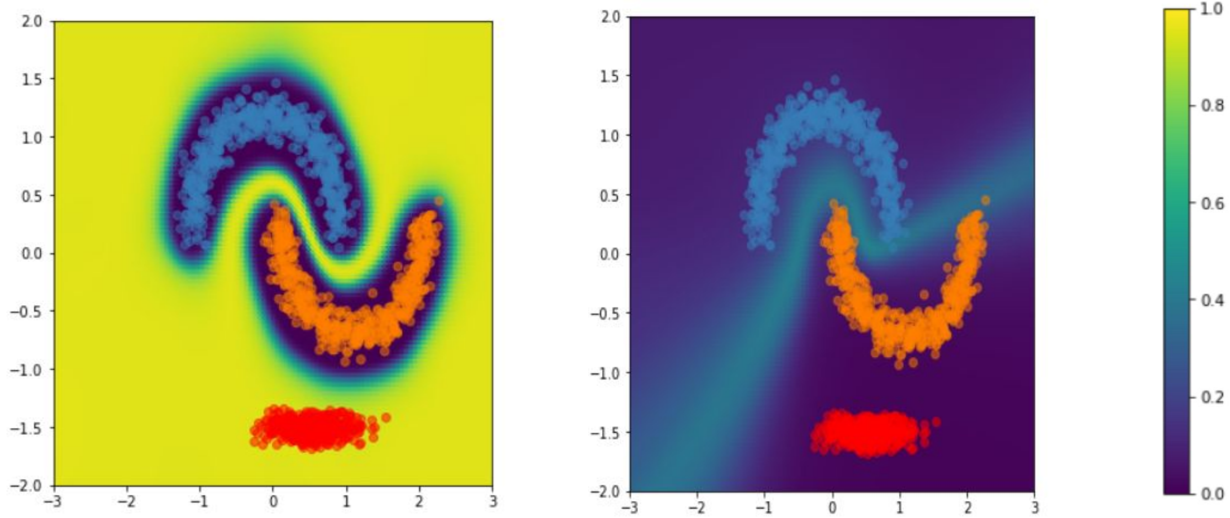


Deep neural networks

Image source: “Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness” Liu et al. 2020

Models assign high confidence predictions to OOD inputs

Google AI
Brain Team



Ideal behavior

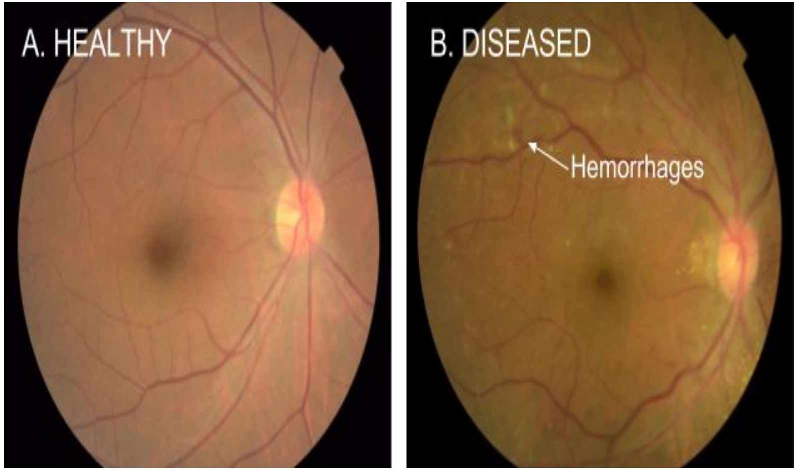
Deep neural networks

Trust model when x* is close to $p_{TRAIN}(x,y)$

Image source: "Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness" [Liu et al. 2020](#)

Applications

Healthcare



Diabetic retinopathy detection from fundus images
[Gulshan et al, 2016]

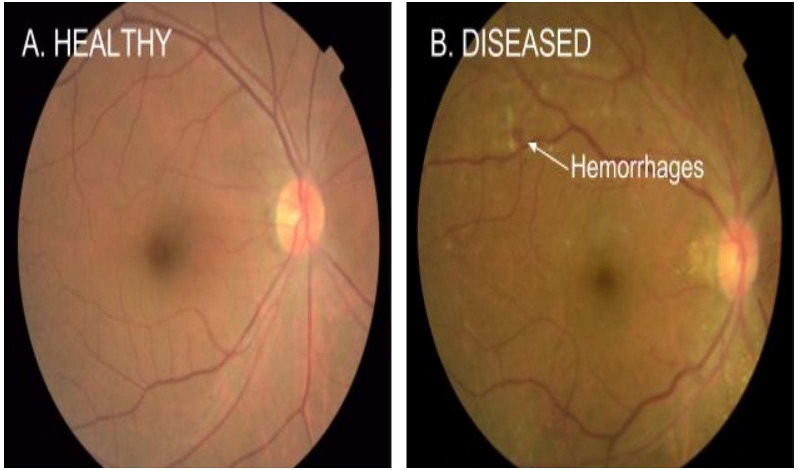
True label

Predicted label		Healthy	Diseased
	Healthy	0	10
	Diseased	1	0

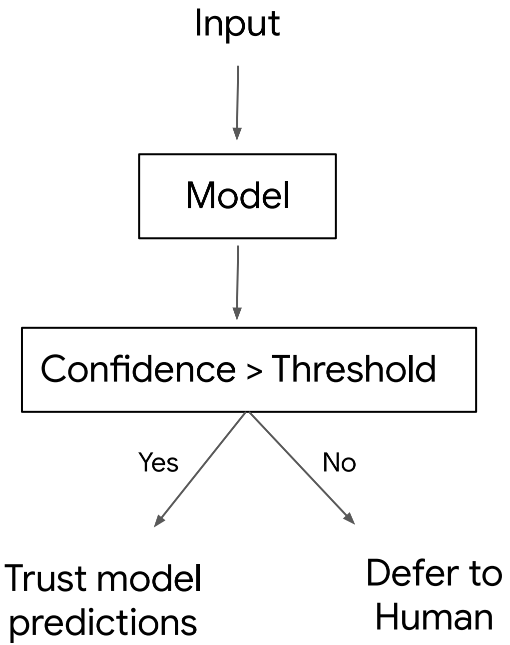
Cost-sensitive decision making

Healthcare

- Use model uncertainty to decide when to trust the model or to defer to a human.
- Reject low-quality inputs.

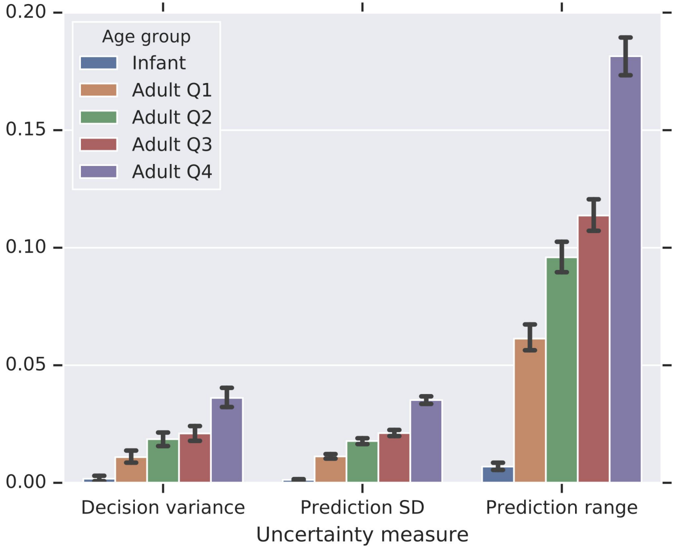


Diabetic retinopathy detection from fundus images
[Gulshan et al, 2016]



Healthcare

- Model accuracy and uncertainty across patient sub-groups



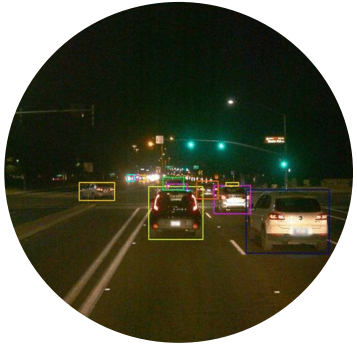
Mortality prediction from electronic health records
[Dusenberry et al, 2020]

Self-driving cars

Dataset shift:

- Time of day / Lighting
- Geographical location (City vs suburban)
- Changing conditions (Weather / Construction)

Daylight



Night

Weather

Construction

Downtown

Suburban

Image credit: Sun et al, Waymo Open Dataset

Open Set Recognition

Google AI
Brain Team

- Example: Classification of genomic sequences

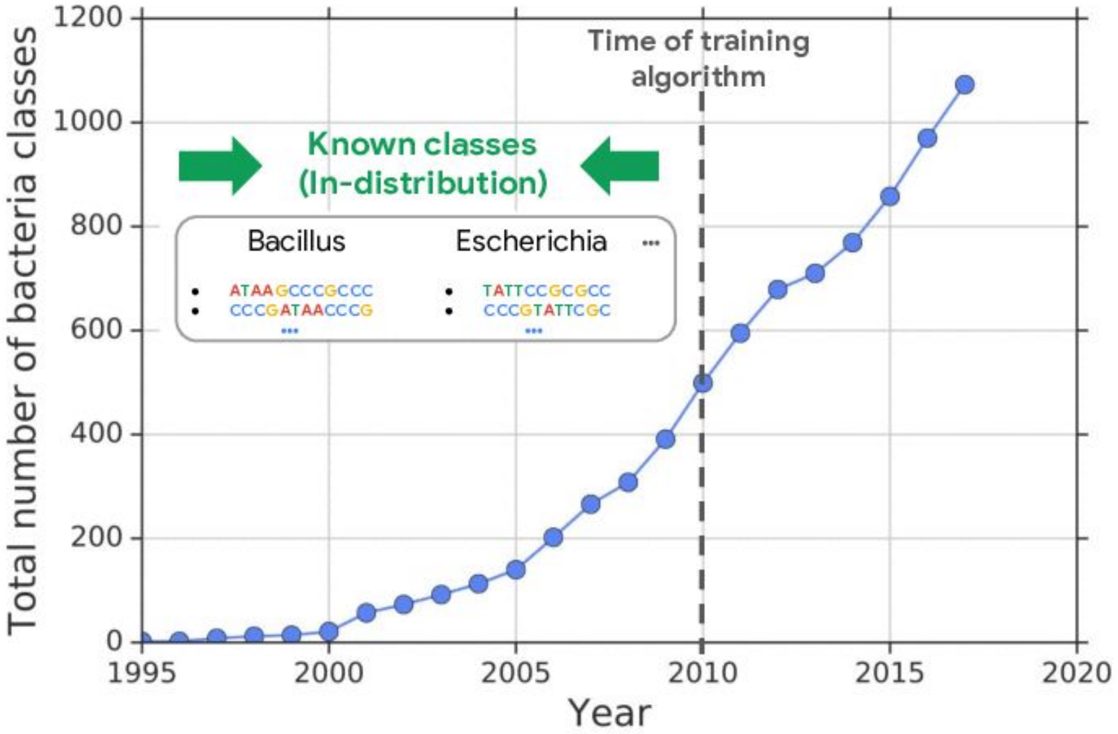


Image source: https://ai.googleblog.com/2019/12/improving-out-of-distribution-detection.html

Open Set Recognition

- Example: Classification of genomic sequences
- High accuracy on known classes is not sufficient
- Need to be able to detect inputs that do not belong to one of the known classes

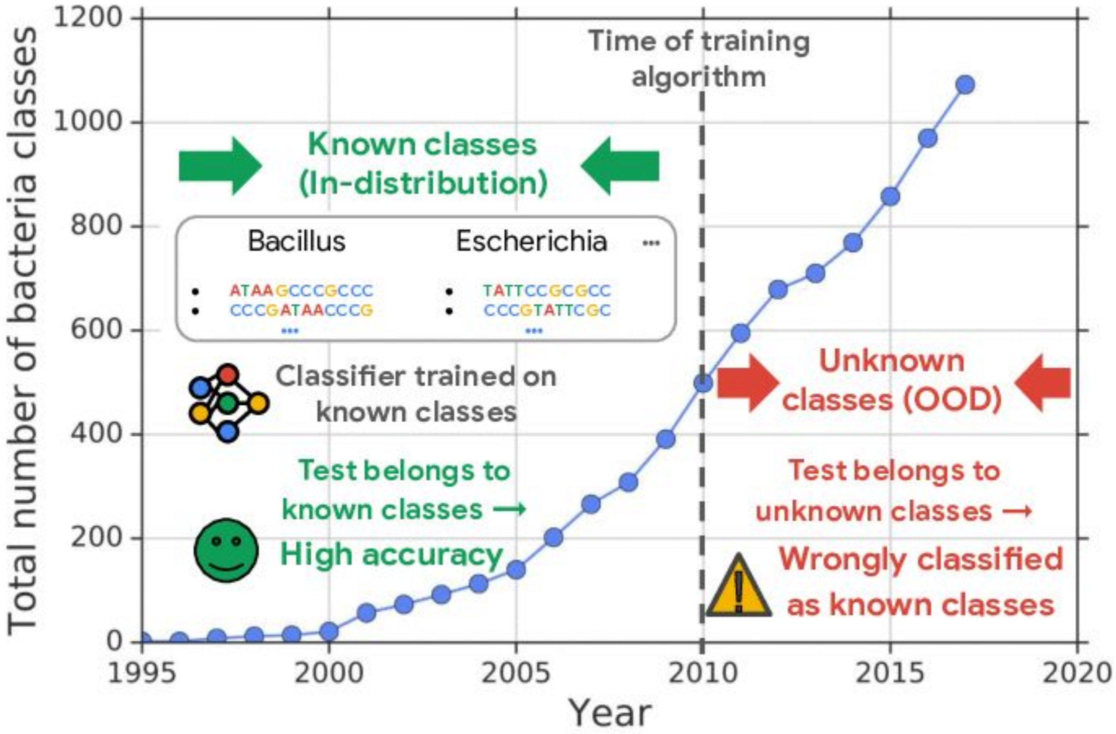


Image source: https://ai.googleblog.com/2019/12/improving-out-of-distribution-detection.html

Conversational Dialog systems

Google AI
Brain Team

- Detecting out-of-scope utterances

Figure 1: Example exchanges between a user (blue, right side) and a task-driven dialog system for personal finance (grey, left side). The system correctly identifies the user's query in (1), but in (2) the user's query is mis-identified as in-scope, and the system gives an unrelated response. In (3) the user's query is correctly identified as out-of-scope and the system gives a fallback response.

Image source: [Larson et al. 2019] "An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction"

Active Learning

- Use model uncertainty to improve data efficiency and model performance in blindspots

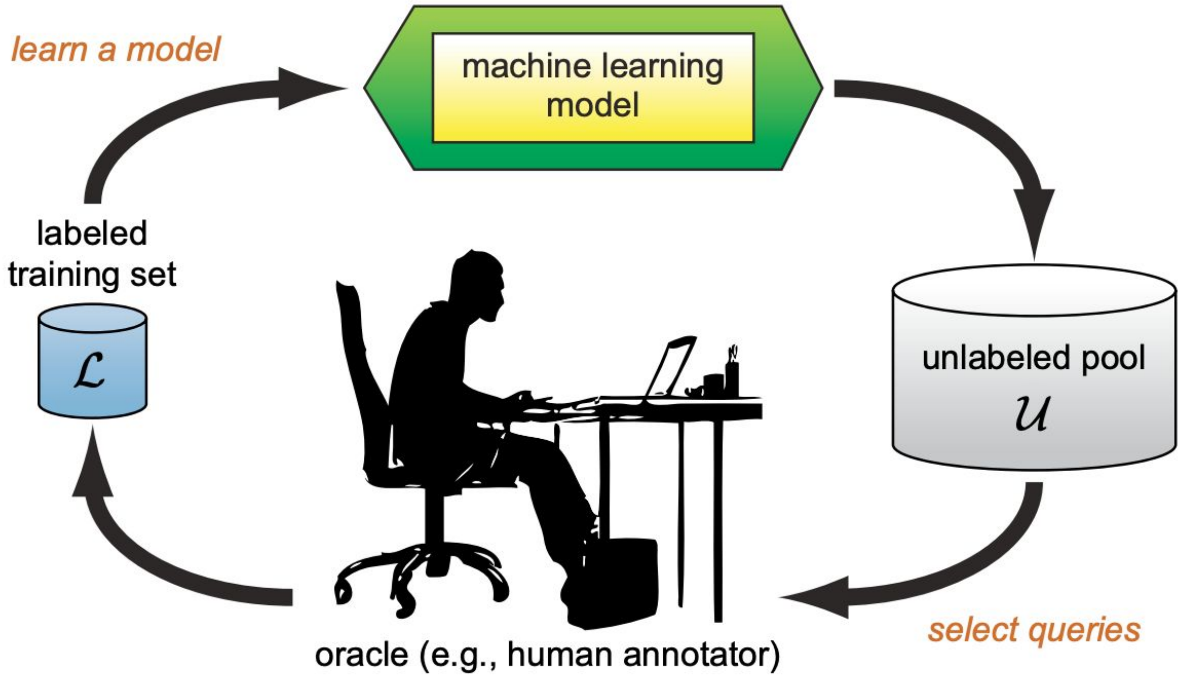
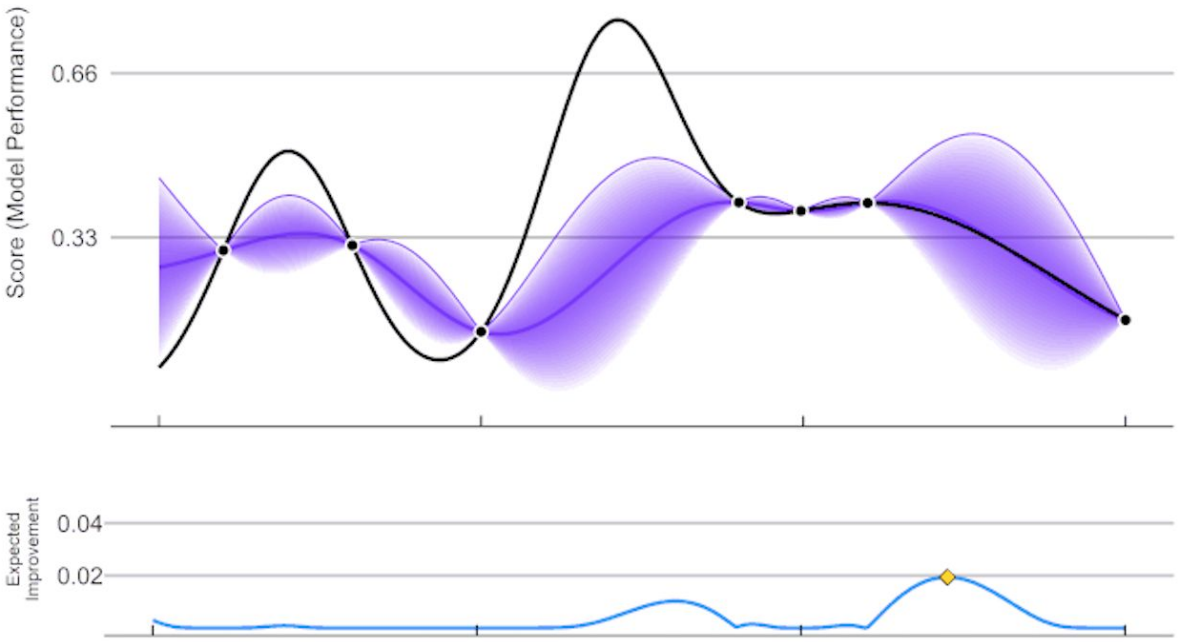


Image source: Active Learning Literature Survey, [Settles 2010]

Bayesian Optimization and Experimental Design

- Which configuration should we explore next?

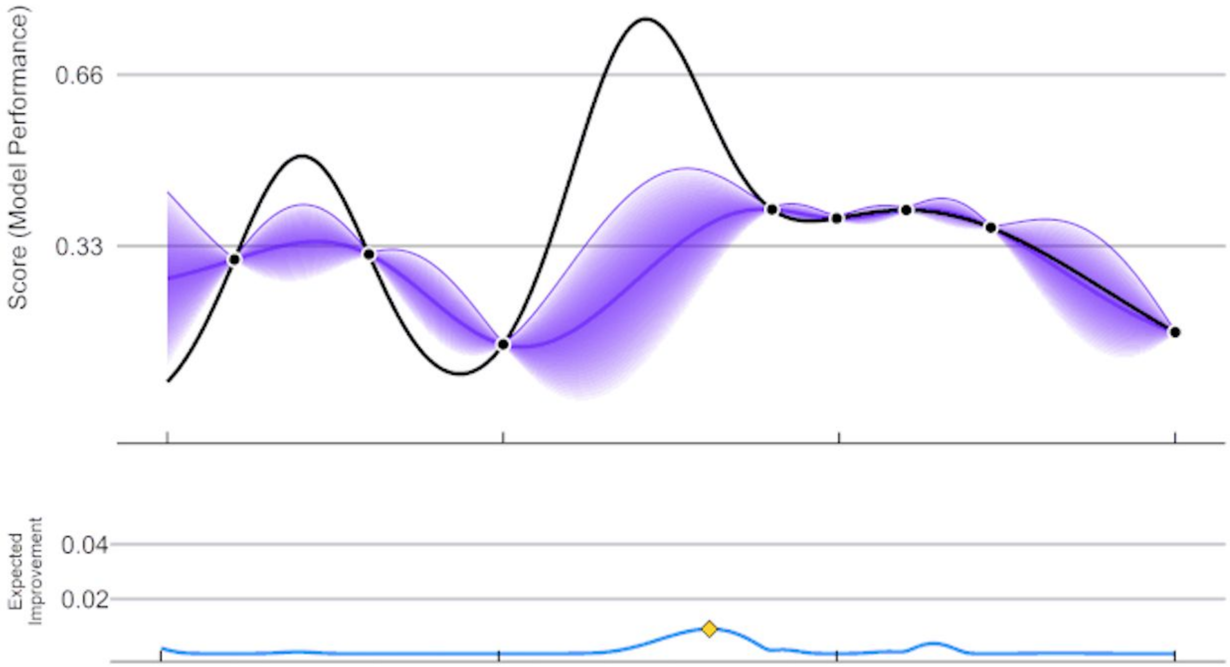


Round 1

Image source: https://en.wikipedia.org/wiki/Bayesian_optimization

Bayesian Optimization and Experimental Design

- Which configuration should we explore next?

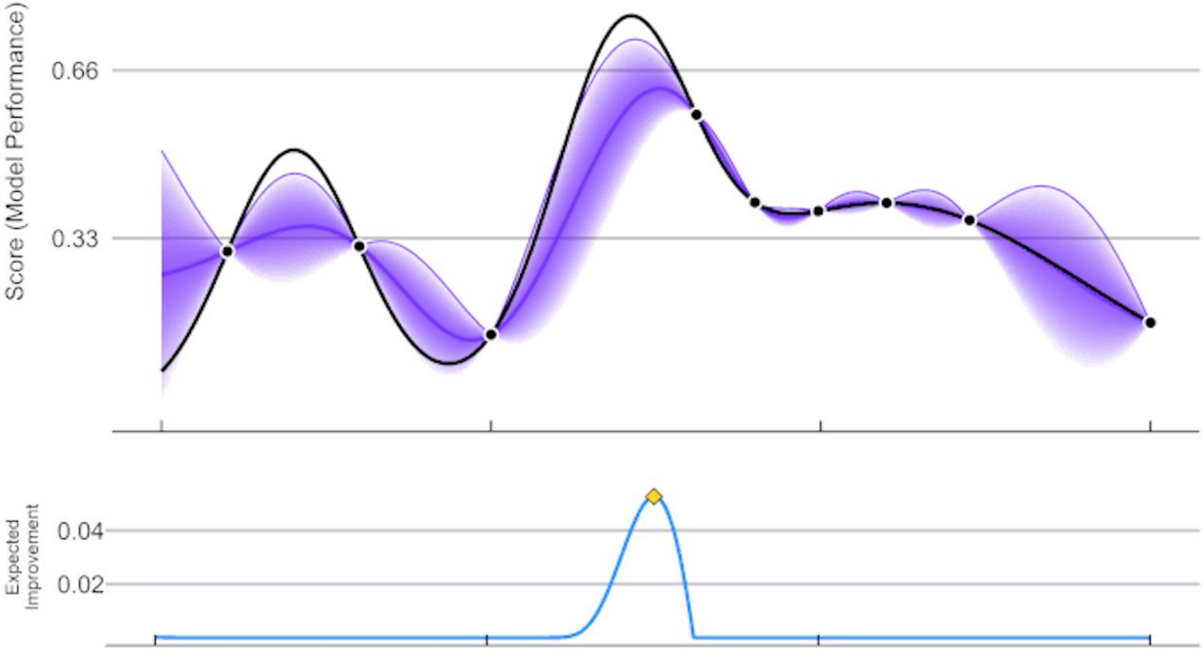


Round 2

Image source: https://en.wikipedia.org/wiki/Bayesian_optimization

Bayesian Optimization and Experimental Design

- Which configuration should we explore next?



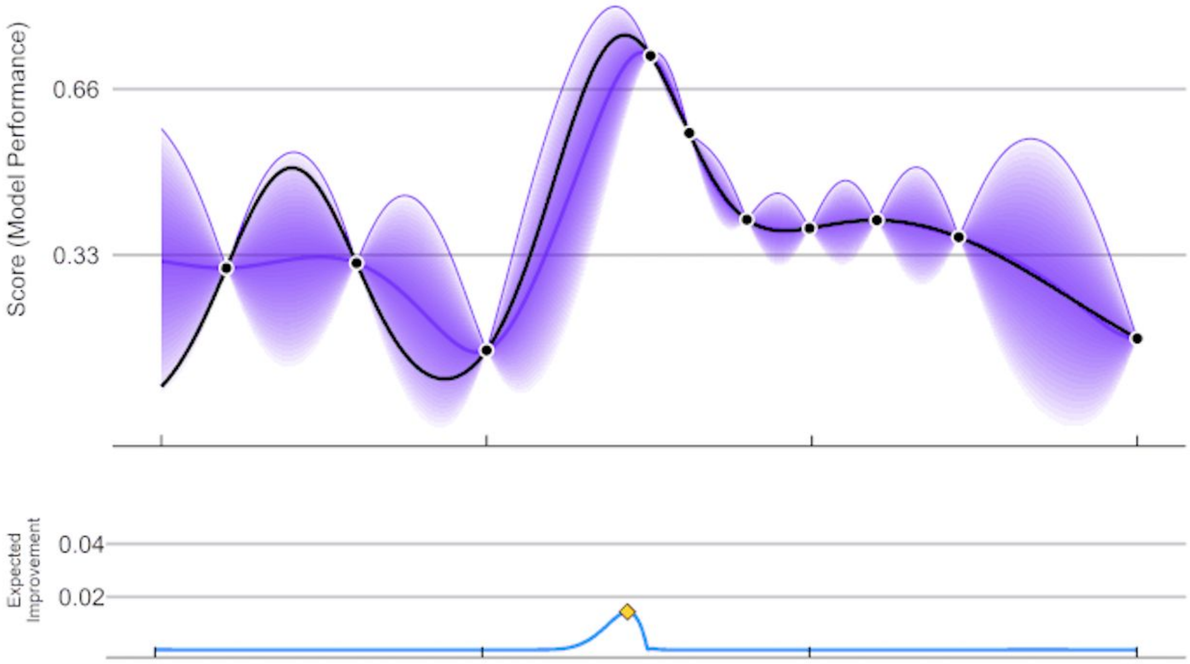
Round 3

Image source: https://en.wikipedia.org/wiki/Bayesian_optimization

Bayesian Optimization and Experimental Design

Google AI
Brain Team

- Which configuration should we explore next?



Round 4

Image source: https://en.wikipedia.org/wiki/Bayesian_optimization

Bayesian Optimization and Experimental Design

Google AI Brain Team

- Hyperparameter optimization and experimental design
  - Used across large organizations and the sciences
- Photovoltaics, chemistry experiments, AlphaGo, batteries, materials design

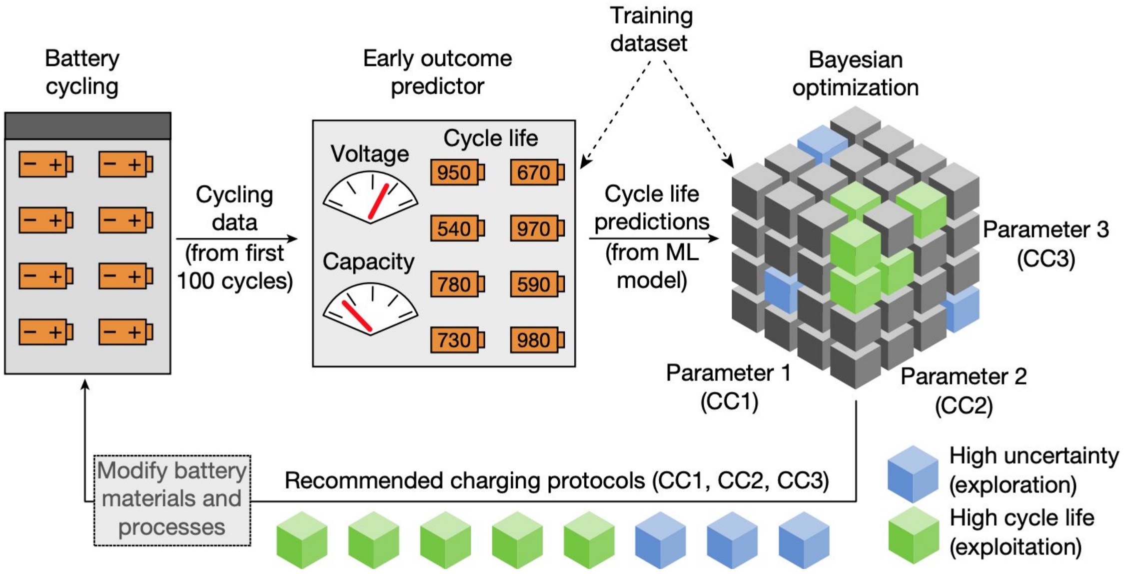


Image source: Attia et al. 2020 Closed-loop optimization of fast-charging protocols for batteries with machine learning

Bandits and Reinforcement Learning

Google AI
Brain Team

Modeling uncertainty is crucial for **exploration vs exploitation** trade-off

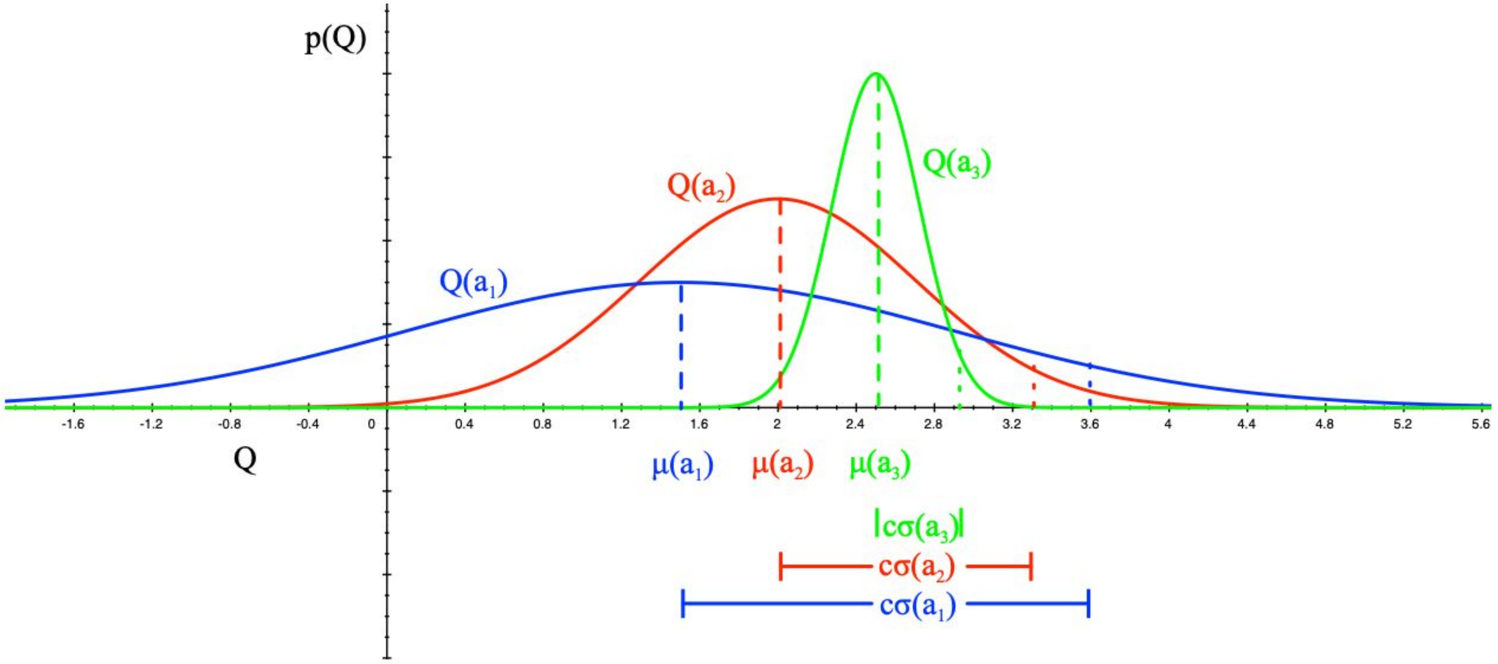


Image source: David Silver's RL course

Bandits and Reinforcement Learning

Google AI
Brain Team

- Decision making with asymmetric losses

$$\ell(\mu) \neq \mathbb{E}_{z \sim N(\mu,\sigma^2)}[\ell(z)]$$

- Distributional Reinforcement learning

- Non-stationarity

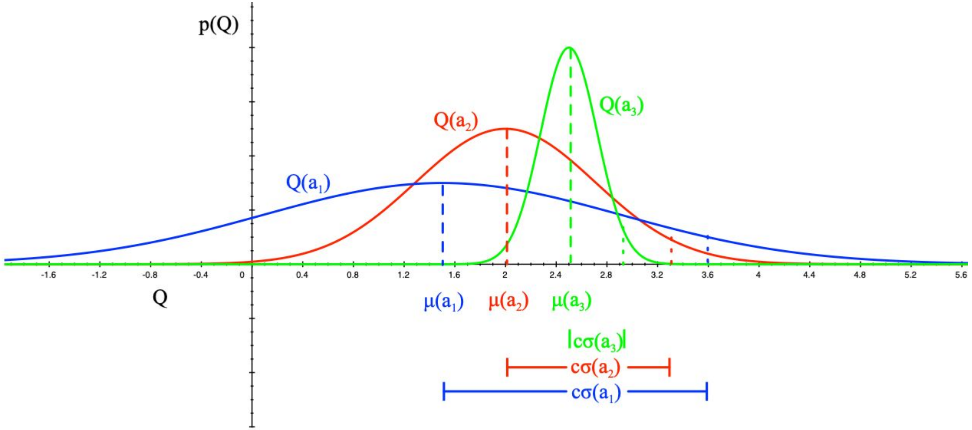


Image source: David Silver's [RL course]

Google AI
Brain Team

Decision making

Safety

Active learning
Lifelong learning

Uncertainty & Out-of-Distribution Robustness

Open-set recognition

Reinforcement learning

Graceful failure

Bayesian optimization

Trustworthy ML

Decision making

Safety

Active learning
Lifelong learning

Uncertainty & Out-of-Distribution Robustness

Open-set
recognition

Reinforcement
learning

Graceful
failure

Bayesian
optimization

Trustworthy
ML

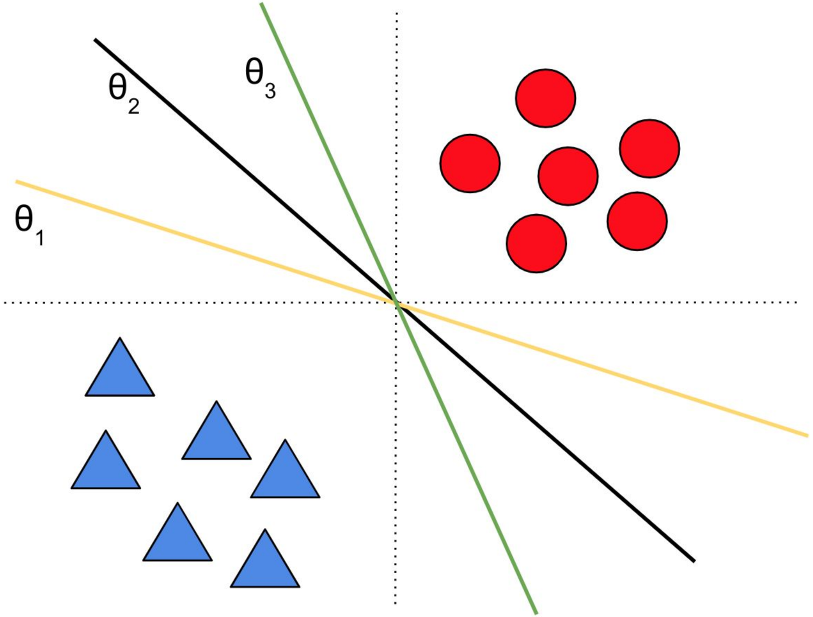
All models are wrong, but ~~some~~ models that know when they are wrong, are useful.

Primer on Uncertainty & Robustness

Sources of uncertainty: *Model uncertainty*

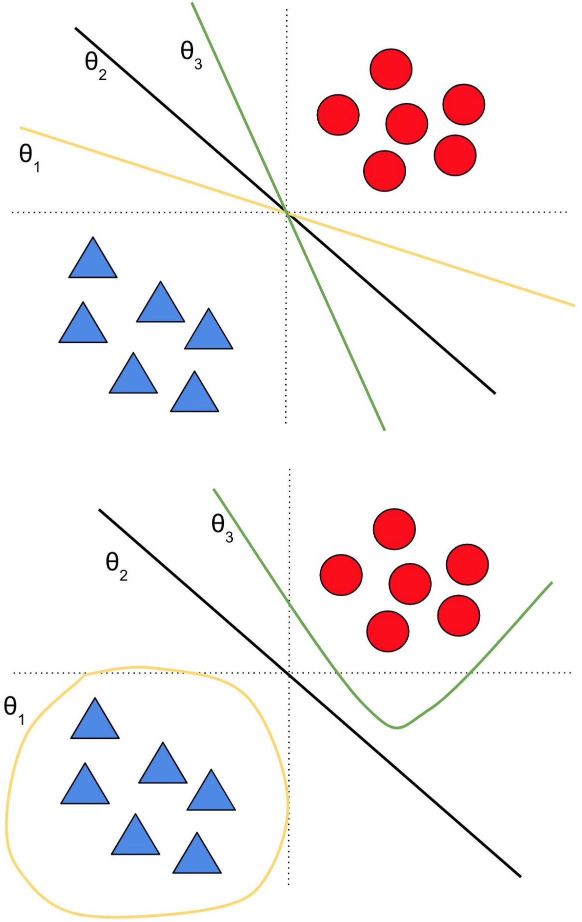
Google AI
Brain Team

- Many models can fit the training data well
- Also known as ***epistemic uncertainty***
- Model uncertainty is "**reducible**"
    - Vanishes in the limit of infinite data (subject to model identifiability)



Sources of uncertainty: *Model uncertainty*

- Many models can fit the training data well
- Also known as ***epistemic uncertainty***
- Model uncertainty is "**reducible**"
  - Vanishes in the limit of infinite data (subject to model identifiability)
- Models can be from same hypotheses class (e.g. linear classifiers in top figure) or belong to different hypotheses classes (bottom figure).



Sources of uncertainty: *Data uncertainty*

- Labeling noise (ex: human disagreement)

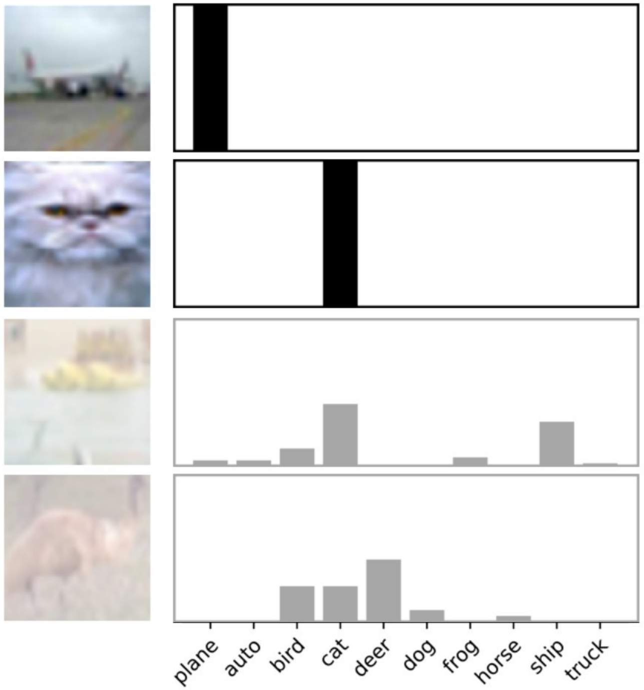


Image source: Battleday et al. 2019 "Improving machine classification using human uncertainty measurements"

Sources of uncertainty: *Data uncertainty*

Google AI
Brain Team

- Labeling noise (ex: human disagreement)

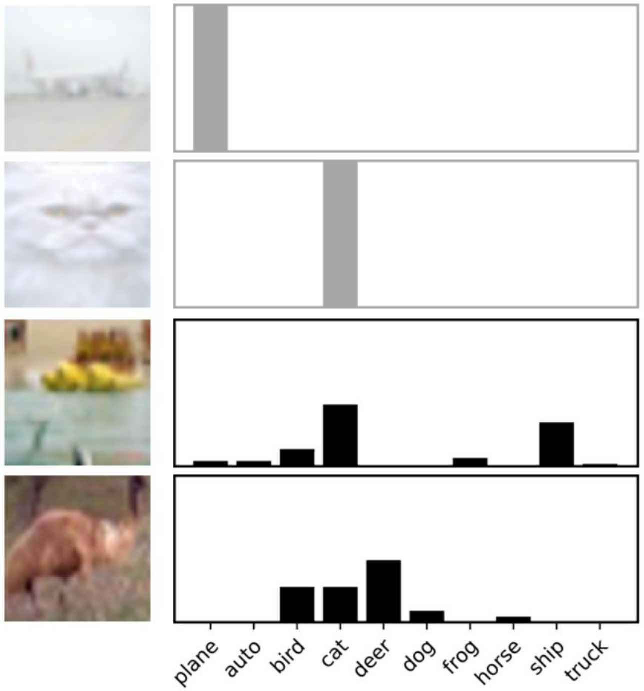


Image source: [Battleday et al. 2019](#) "Improving machine classification using human uncertainty measurements"

Sources of uncertainty: *Data uncertainty*

Google AI Brain Team

- Labeling noise (ex: human disagreement)
- Measurement noise (ex: imprecise tools)
- *Missing* data (ex: partially observed features, unobserved confounders)
- Also known as ***aleatoric uncertainty***
- Data uncertainty is "**irreducible***"
  - Persists even in the limit of infinite data
  - *Could be reduced with additional features/views

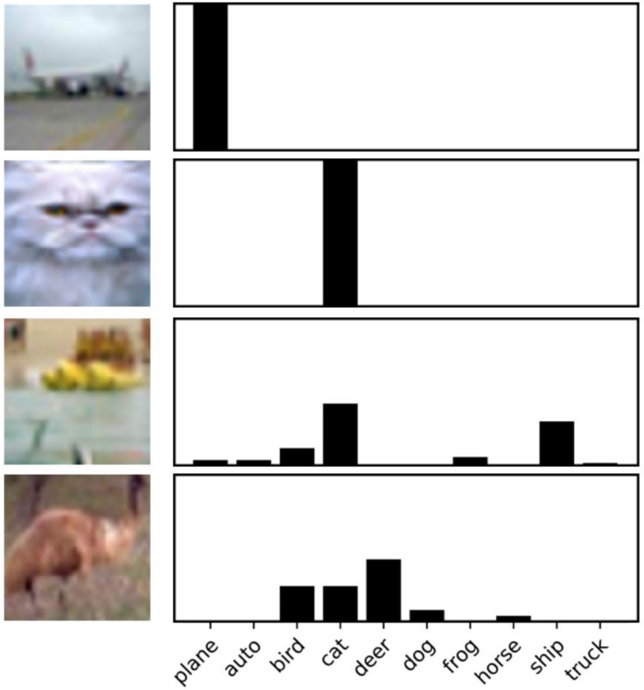


Image source: [Battleday et al. 2019](#) "Improving machine classification using human uncertainty measurements"

How do we measure the quality of uncertainty?

Calibration Error = |Confidence - Accuracy|

predicted probability of correctness (Confidence)

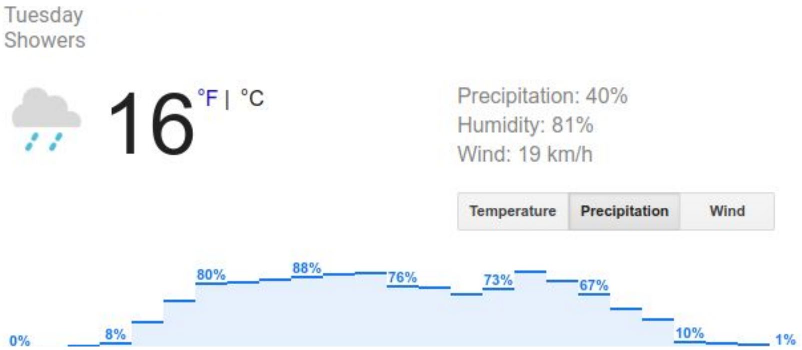
observed frequency of correctness (Accuracy)

How do we measure the quality of uncertainty?

Calibration Error = |Confidence - Accuracy|

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

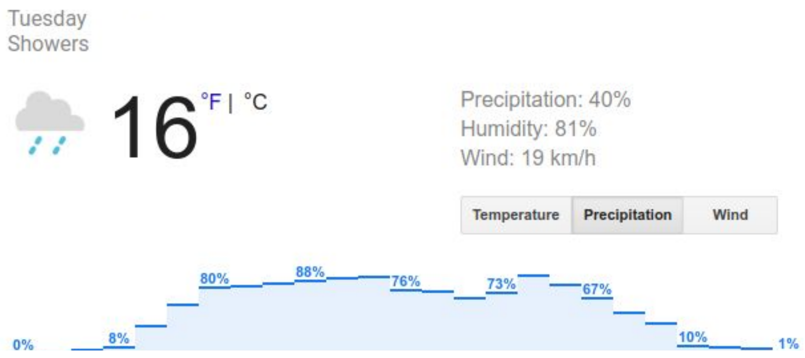


How do we measure the quality of uncertainty?

Calibration Error = |Confidence - Accuracy|

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



Intuition: For regression, calibration corresponds to coverage in a confidence interval.

How do we measure the quality of uncertainty?

Expected Calibration Error [Naeini+ 2015]:

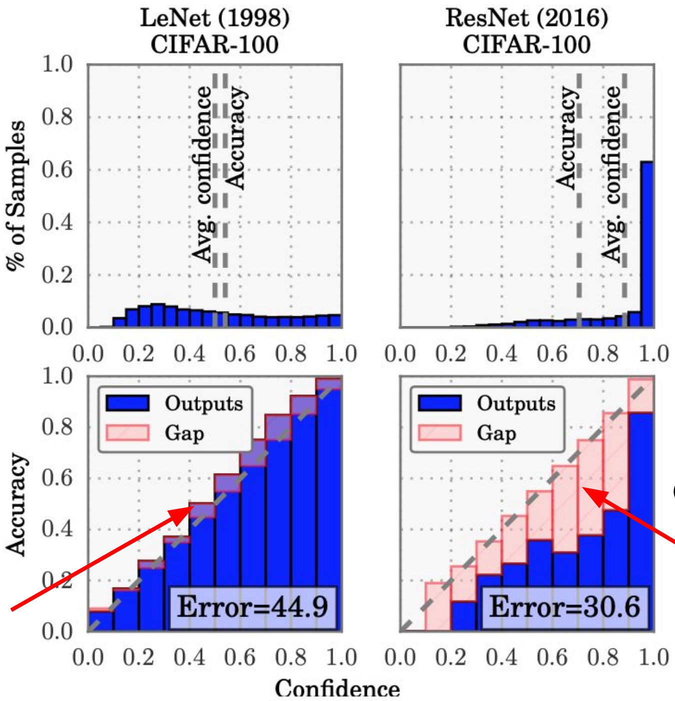
$$\mathrm{ECE} = \sum_{b=1}^{B} \frac{n_b}{N} \left|\mathrm{acc}(b) - \mathrm{conf}(b)\right|$$

- Bin the probabilities into B bins.
- Compute the within-bin accuracy and within-bin predicted confidence.
- Average the calibration error across bins (weighted by number of points in each bin).

How do we measure the quality of uncertainty?

Expected Calibration Error [Naeini+ 2015]:

$$\mathrm{ECE} = \sum_{b=1}^{B} \frac{n_b}{N} \left|\mathrm{acc}(b) - \mathrm{conf}(b)\right|$$



Confidence < Accuracy

=> Underconfident

Confidence > Accuracy

=> Overconfident

Image source: Guo+ 2017 "On calibration of modern neural networks"

How do we measure the quality of uncertainty?

Expected Calibration Error [Naeini+ 2015]:

$$\mathrm{ECE} = \sum_{b=1}^{B} \frac{n_b}{N} \left|\mathrm{acc}(b) - \mathrm{conf}(b)\right|$$

Note: Does **not** reflect **accuracy**.

Predicting class frequency p(y=1) = 0.3 for all the inputs achieves perfect calibration.

True label	0	0	0	0	0	0	0	1	1	1	Accurate?	Calibrated?
Model prediction	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	❌	✔️

How do we measure the quality of uncertainty?

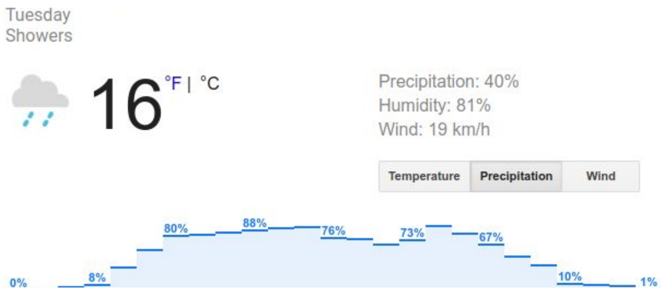
Proper scoring rules [Gneiting & Raftery 2007]

- Negative Log-Likelihood (NLL)
  - Also known as *cross-entropy*
  - Can overemphasize tail probabilities

- Brier Score
  - Quadratic penalty (bounded range [0,1] unlike log).

$$\mathrm{BS} = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} \left[p(y|\mathbf{x}_n, \theta) - \delta(y - y_n) \right]^2$$

  - Can be numerically unstable to optimize.

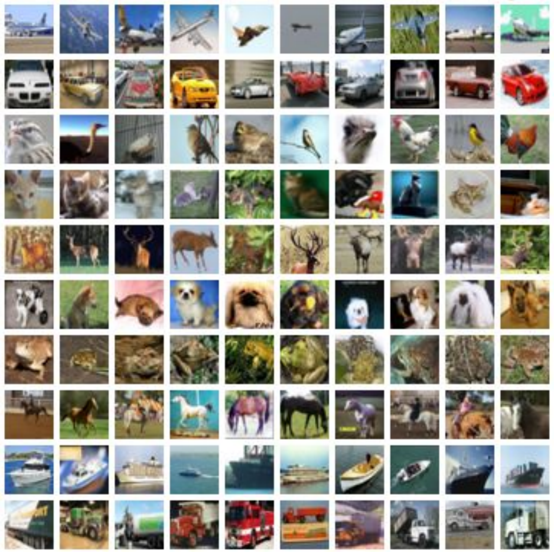


How do we measure the quality of uncertainty?

Google AI
Brain Team

Evaluate model on **out-of-distribution (OOD) inputs** which do not belong to any of the existing classes

- Max confidence
- Entropy of p(y|x)



CIFAR-10 (i.i.d test inputs)

SVHN (o.o.d test inputs)

CIFAR-10 classifier

Confidence on i.i.d inputs **>** Confidence on o.o.d inputs **?**

Google AI
Brain Team

How do we measure the quality of robustness?

Measure generalization to a *large collection of real-world shifts*. A large collection of tasks encourages *general robustness to shifts* (ex: GLUE for NLP).

- Novel textures in object recognition.
- Covariate shift (e.g. corruptions).
- Different sub-populations (e.g. geographical location).

Different renditions
(ImageNet-R)

Nearby video frames
(ImageNet-Vid-Robust, YTBB-Robust)

Multiple objects and poses
(ObjectNet)

Coffee Break (15 mins) ☕

Check out the Q&A.

1. Why uncertainty & robustness?

2. Foundations.

3. Recent advances.

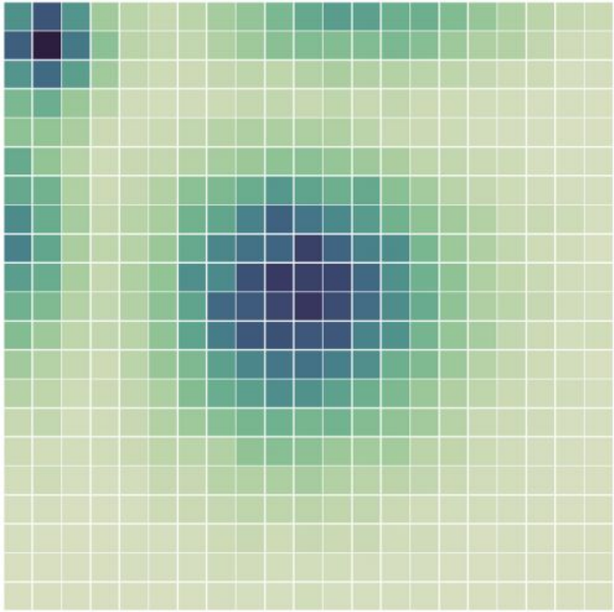
Fundamentals to Uncertainty & Robustness Methods

Neural Networks with SGD

Google AI
Brain Team

Nearly all models find a single setting of parameters to maximize the probability conditioned on data.

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{x}, \mathbf{y})$$
$$= \arg\min_{\boldsymbol{\theta}} -\log p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta})$$
$$=^* \arg\min_{\boldsymbol{\theta}} \sum_k \mathbf{y}_k \log \mathbf{p}_k + \boldsymbol{\lambda} \|\boldsymbol{\theta}\|^2$$



Special case: softmax cross entropy with L2 regularization. Optimize with SGD!

Image source: Ranganath+ 2016

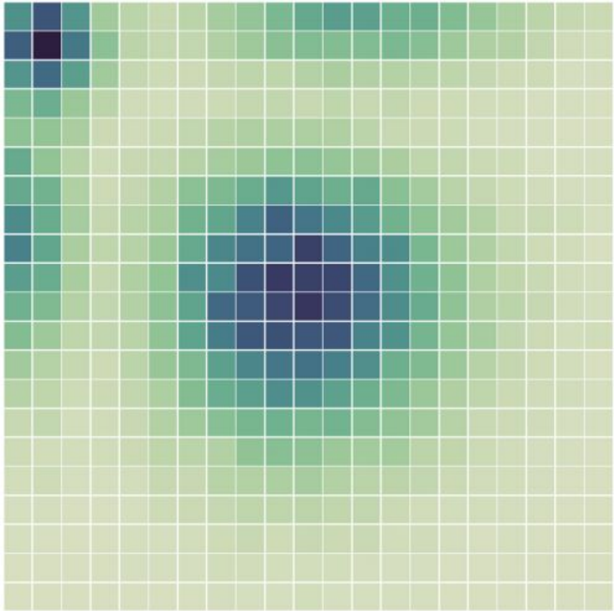
Neural Networks with SGD

Google AI Brain Team

Nearly all models find a single setting of parameters to maximize the probability conditioned on data.

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{x}, \mathbf{y})$$
$$= \arg\min_{\boldsymbol{\theta}} -\log p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta})$$

Data uncertainty



Special case: softmax cross entropy with L2 regularization. Optimize with SGD!

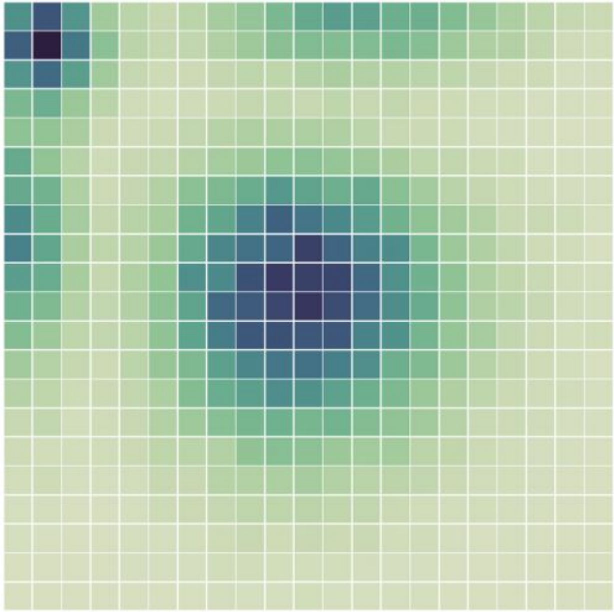
Image source: Ranganath+ 2016

Neural Networks with SGD

Google AI Brain Team

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{x}, \mathbf{y})$$

Problem: results in just one prediction per example
No model uncertainty



How do we get uncertainty?

- Probabilistic approach
  - Estimate a full distribution for $p(\boldsymbol{\theta} \mid \mathbf{x}, \mathbf{y})$
- Intuitive approach: Ensembling
  - Obtain multiple good settings for $\boldsymbol{\theta}^*$

Image source: Ranganath+ 2016

Probabilistic Machine Learning

Model: A probabilistic model is a joint distribution of outputs **y** and parameters $\boldsymbol{\theta}$ given inputs **x**.

$$p(\mathbf{y}, \boldsymbol{\theta} \,|\, \mathbf{x})$$

Training time: Calculate the **posterior**, the conditional distribution of parameters given observations.

$$p(\boldsymbol{\theta} \,|\, \mathbf{x}, \mathbf{y}) = \frac{p(\mathbf{y}, \boldsymbol{\theta} \,|\, \mathbf{x})}{p(\mathbf{y} \,|\, \mathbf{x})} = \frac{p(\mathbf{y} \,|\, \mathbf{x}) p(\boldsymbol{\theta})}{\int p(\mathbf{y}, \boldsymbol{\theta} \,|\, \mathbf{x}) \, \mathrm{d}\boldsymbol{\theta}}$$

Prediction time: Compute the likelihood given parameters, each parameter configuration of which is weighted by the posterior.

$$p(\mathbf{y} \,|\, \mathbf{x}, \mathcal{D}) = \int p(\mathbf{y} \,|\, \mathbf{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \,|\, \mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} \approx \frac{1}{S} \sum_{s=1}^{S} p(\mathbf{y} \,|\, \mathbf{x}, \boldsymbol{\theta}^{(s)})$$

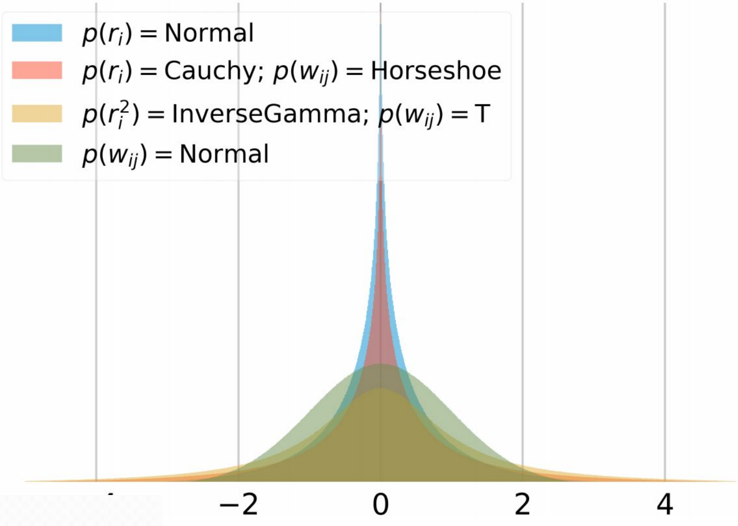
[Murphy 2012]

Bayesian Neural Networks

Google AI Brain Team

Bayesian neural nets specify a distribution over neural network predictions.

This is done by specifying a distribution over neural network weights $p(\boldsymbol{\theta})$.



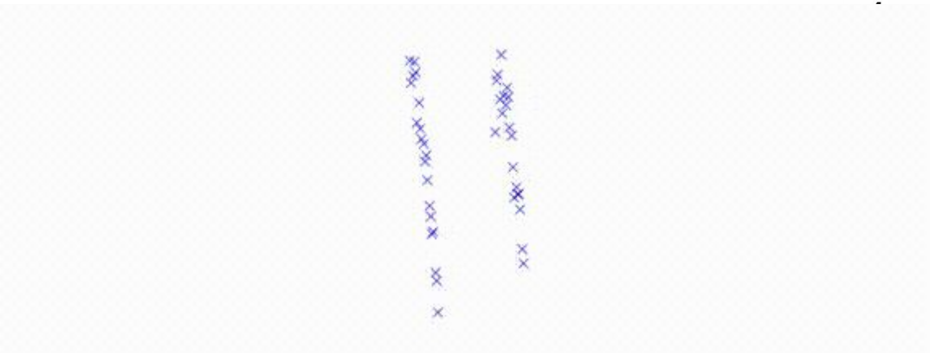


Image source: Dusenberry+ 2020

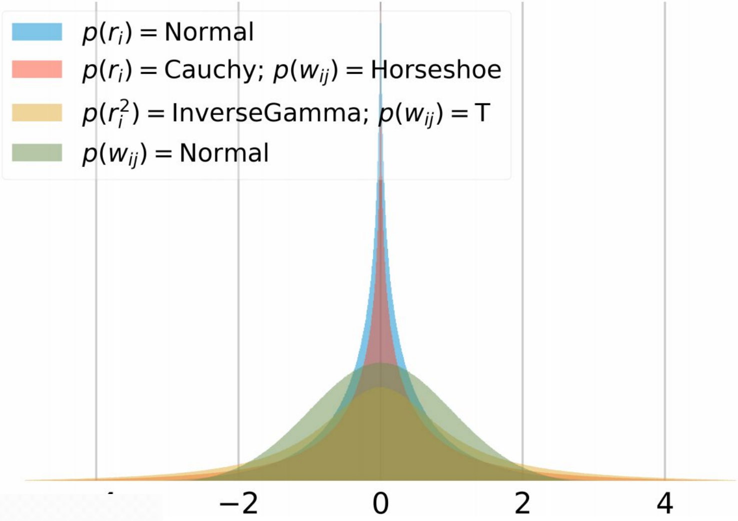
Bayesian Neural Networks

Google AI
Brain Team

Bayesian neural nets specify a distribution over neural network predictions.

This is done by specifying a distribution over neural network weights $p(\boldsymbol{\theta})$.

We can reason about uncertainty in models away from the data!



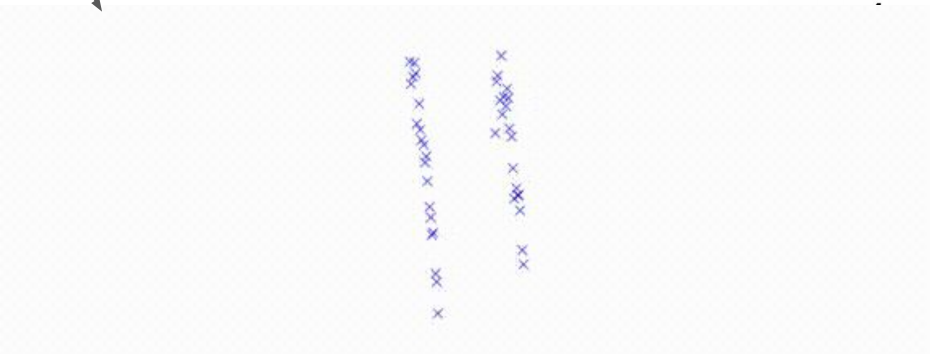
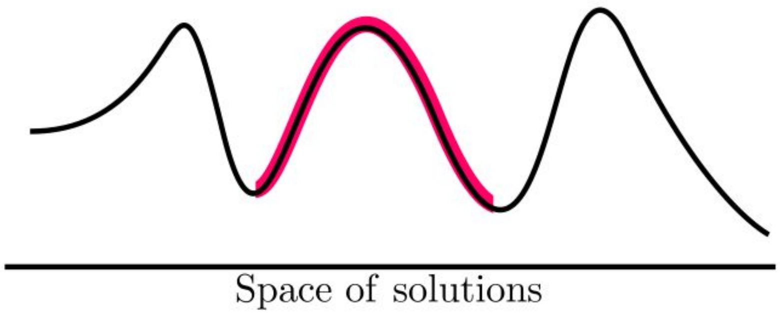


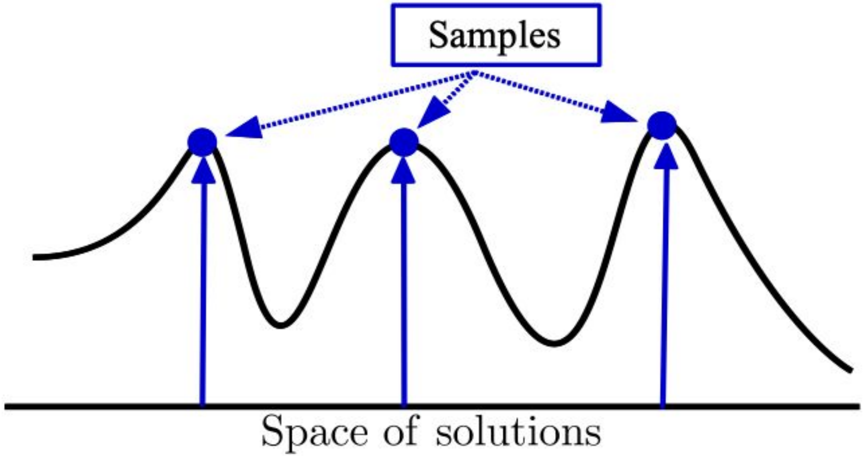
Image source: [Dusenberry+ 2020]

Approximating the posterior

$p(\boldsymbol{\theta} \mid \mathcal{D})$ is multimodal and complex, so how do we estimate and represent it?



Local approximations

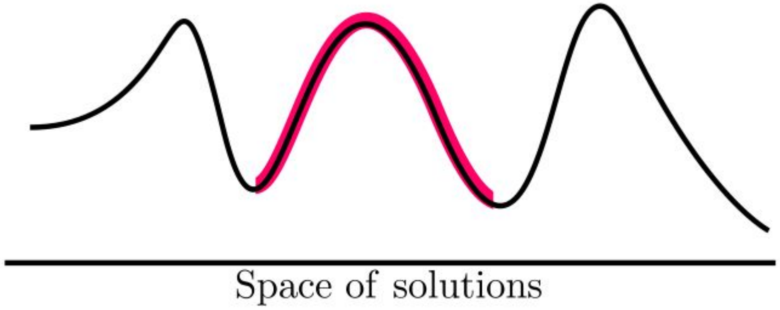


Sampling

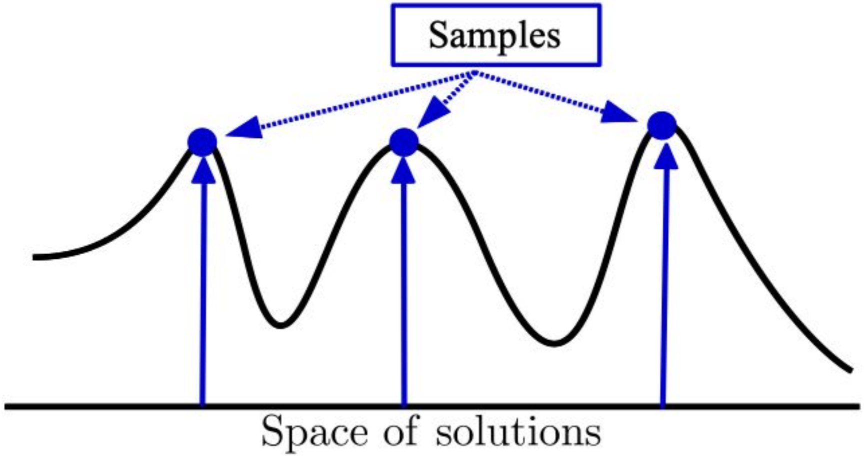
- Locally, covering one mode well e.g. with a simpler distribution $q(\boldsymbol{\theta}; \boldsymbol{\lambda})$
  - Variational inference
  - Laplace approximation

Approximating the posterior

$p(\boldsymbol{\theta} \mid \mathcal{D})$ is multimodal and complex, so how do we estimate and represent it?



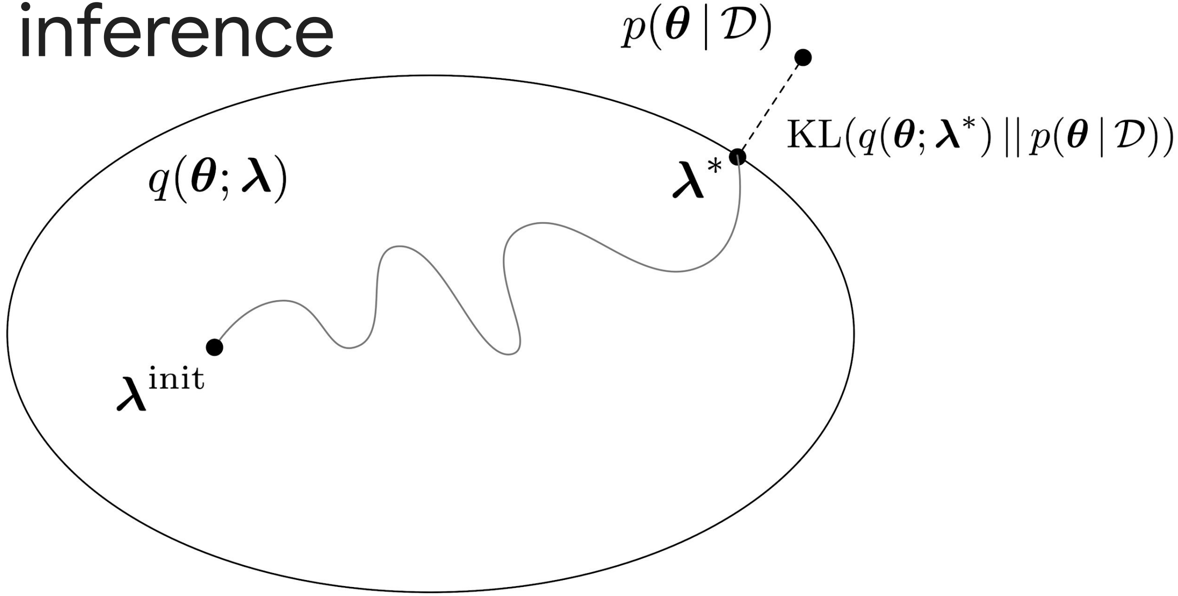
Local approximations



Sampling

- Summarize using samples
  - MCMC
  - Hamiltonian Monte Carlo
  - Stochastic Gradient Langevin Dynamics

Variational inference



- VI casts posterior inference as an optimization problem.
- Posit a **family of variational distributions** over $\boldsymbol{\theta}$ such as mean-field,

$$q(\boldsymbol{\theta}; \boldsymbol{\lambda}) = \prod_i q(\boldsymbol{\theta}_i; \boldsymbol{\lambda}_i)$$

- Optimize a **divergence measure** (such as KL) with respect to $\boldsymbol{\lambda}$ to be close to the posterior .

Image source: Blei+ 2016. NeurIPS tutorial

Bayesian Neural Networks with SGD

The loss function in variational inference is

$$\mathcal{L}(\boldsymbol{\lambda}) = -\mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\lambda})}[\log p(\mathbf{y} \,|\, \mathbf{x}, \boldsymbol{\theta})] + \mathrm{KL}(q(\boldsymbol{\theta}; \boldsymbol{\lambda}) \,\|\, p(\boldsymbol{\theta}))$$

Sample from **q** to Monte Carlo estimate the expectation. Take gradients for SGD.

Likelihood view. The negative of the loss is a lower bound to the marginal likelihood.

$$-\mathcal{L}(\boldsymbol{\lambda}) \leq \log p(\mathbf{y} \,|\, \mathbf{x}) \qquad \text{for all } \boldsymbol{\lambda} \in \boldsymbol{\Lambda}$$

Code length view. Minimize the # of bits to explain the data, while trying not to pay many bits when deviating from the prior.

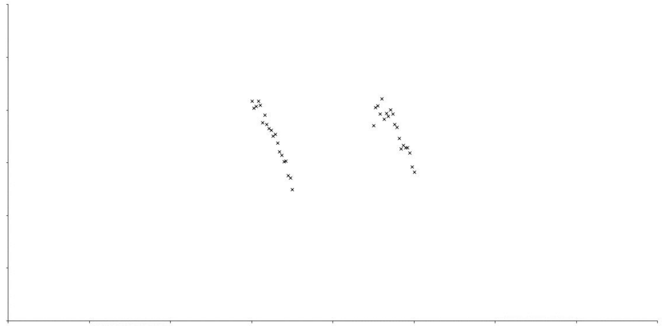
Check out [Approximate Inference Symposium, Jan 2021]

Infinite Width Bayesian Deep Networks are Gaussian Processes

- A specific parameterization of an NN defines a function

- Thus a BNN defines a *distribution* over functions
  - Induced by the distribution over weights $p(\boldsymbol{\theta} \mid \mathbf{x}, \mathbf{y})$

- How do we reason about this distribution in general?



Visualizing the distribution over functions

- It turns out that this corresponds to a known model class in an important case
  - In the limit of infinite width converges to a *Gaussian Process* [Neal 1994]

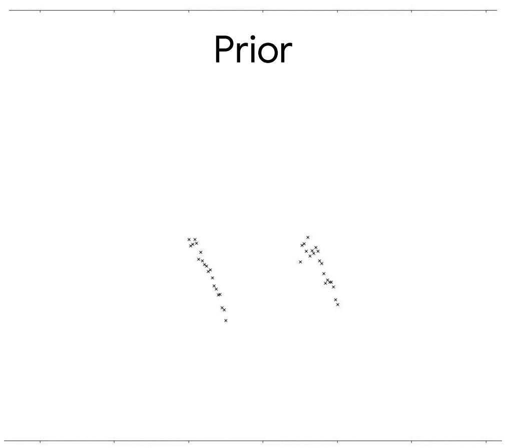
Gaussian Processes

We can compute the integral $p(y|x, \mathcal{D}) = \int p(y|x, \boldsymbol{\theta})\, p(\boldsymbol{\theta}|\mathcal{D})\, \mathrm{d}\boldsymbol{\theta}$ analytically!

Under Gaussian likelihood + prior and
in the limit of infinite basis functions (e.g. hidden units) -> GP

The result is a flexible distribution over functions

- Specified now by a covariance function over examples $K(X, X)$
  - Covariance over the basis functions
  - Familiar with the kernel trick?



Posterior

See [Rasmussen & Williams 2006]

Google AI
Brain Team

Gaussian Processes

We can compute the integral $p(y|x,\mathcal{D}) = \int p(y|x,\boldsymbol{\theta})\, p(\boldsymbol{\theta}|\mathcal{D})\, \mathrm{d}\boldsymbol{\theta}$ analytically!

Under Gaussian likelihood + prior and
in the limit of infinite basis functions (e.g. hidden units) -> GP

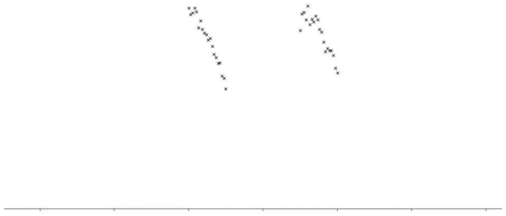
The result is a flexible distribution over functions

Prior

- Specified now by a covariance function over examples $K(X,X)$

- Get a posterior on functions conditioned on data

$$\mathbf{f}_*|X,\mathbf{y},X_* \sim \mathcal{N}\big(\bar{\mathbf{f}}_*,\ \mathrm{cov}(\mathbf{f}_*)\big), \text{ where}$$
$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}[\mathbf{f}_*|X,\mathbf{y},X_*] = K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}\mathbf{y},$$
$$\mathrm{cov}(\mathbf{f}_*) = K(X_*,X_*) - K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}K(X,X_*)$$



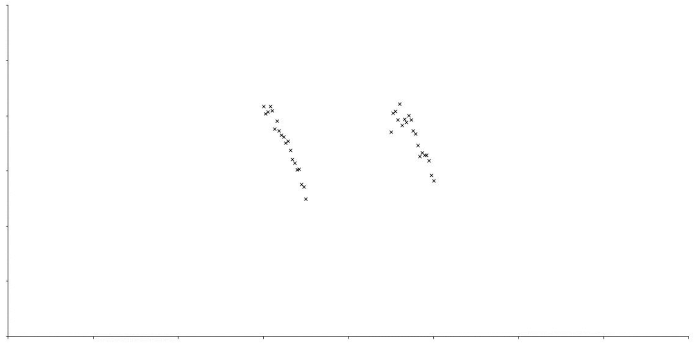
Posterior

See [Rasmussen & Williams 2006]

Infinite Width Deep Neural Networks are Gaussian Processes

- Renewed interest after [Neal 1994]
  - Deep Neural Networks as Gaussian Processes, Lee 2018
  - Gaussian process behaviour in wide deep neural networks, Matthews 2018
  - + many more.

- Allows us to reason about the behavior of neural networks in exciting new ways
  - Without nuances of training, hidden units, etc.
  - e.g. generalization properties, Adlam 2020



NNGP Posterior

- It turns out they are well calibrated!
  - Exploring the Uncertainty Properties of Neural Networks' Implicit Priors in the Infinite-Width Limit, Adlam+ 2020

- Want to play around with infinitely wide networks? neural tangents library

Ensemble Learning

- A prior distribution often involves the complication of approximate inference.
- *Ensemble learning* offers an alternative strategy to aggregate the predictions over a collection of models.
- Often winner of competitions!
- There are two considerations: the collection of models to ensemble; and the aggregation strategy.

Popular approach is to average predictions of independently trained models, forming a mixture distribution.

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{K} \sum_{k=1}^{K} p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}_k)$$

Many approaches exist: bagging, boosting, decision trees, stacking.

[Dietterich 2000]

Bayes vs Ensembles: What’s the difference?

Both aggregate predictions over a collection of models. There are two core distinctions.

The space of models.

Bayes posits a prior that weighs different probability to different functions, and over an infinite collection of functions.

Ensembles weigh functions equally a priori and use a finite collection

Model aggregation.

Bayesian models apply averaging, weighted by the posterior.

Ensembles can apply any strategy and have non-probabilistic interpretations.

In the community, it’s popular to cast one as a “special case” of the other, under trivial settings. However, Bayes and ensembles are critically different mindsets.

Bayesian model averaging is not model combination. Minka 2002
Bayesian Deep Ensembles via the Neural Tangent Kernel. He, Lakshminarayanan, Teh, NeurIPS 2020

Challenges with Bayes

Lots of recent methods tweak Bayes rule slightly

- Tempering the posterior or downweighting the prior
- Making it unclear what the model actually is

The two objectives of VI complicate the dynamics of training

- New heuristics to train these models
  - Initialization, etc.

Bayes makes sense when the model is well specified

- This remains a challenge for a lot of deep networks
- Sub-optimal when the model is misspecified [Masegosa 2020]

How Good is the Bayes Posterior in Deep Neural Networks Really? Wenzel+ 2020

Simple Baselines

Simple Baseline: Recalibration

For classification, modify softmax probabilities post-hoc.

Temperature Scaling.

1. Parameterize output layer with scalar T.

$$p(y_i|x) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

2. Minimize loss with respect to T on a separate "recalibration" dataset.

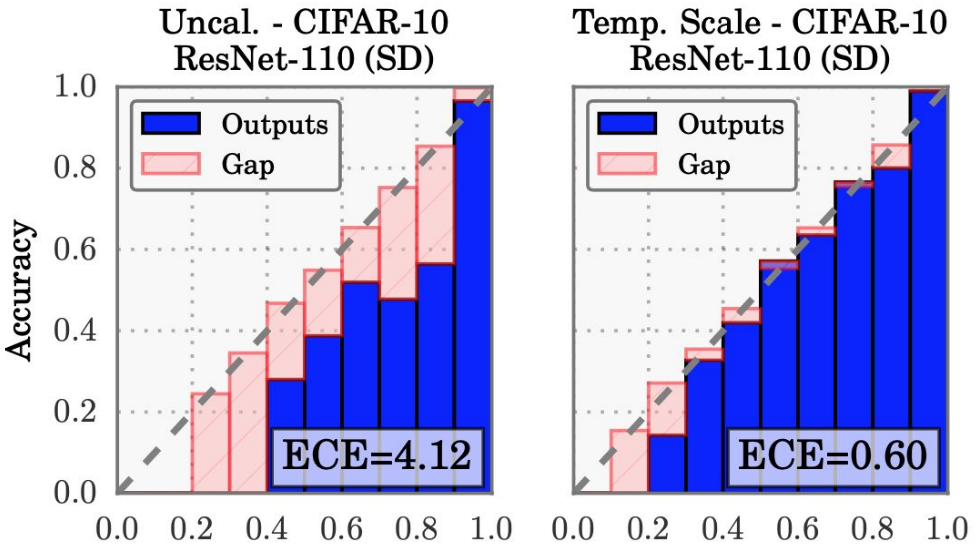
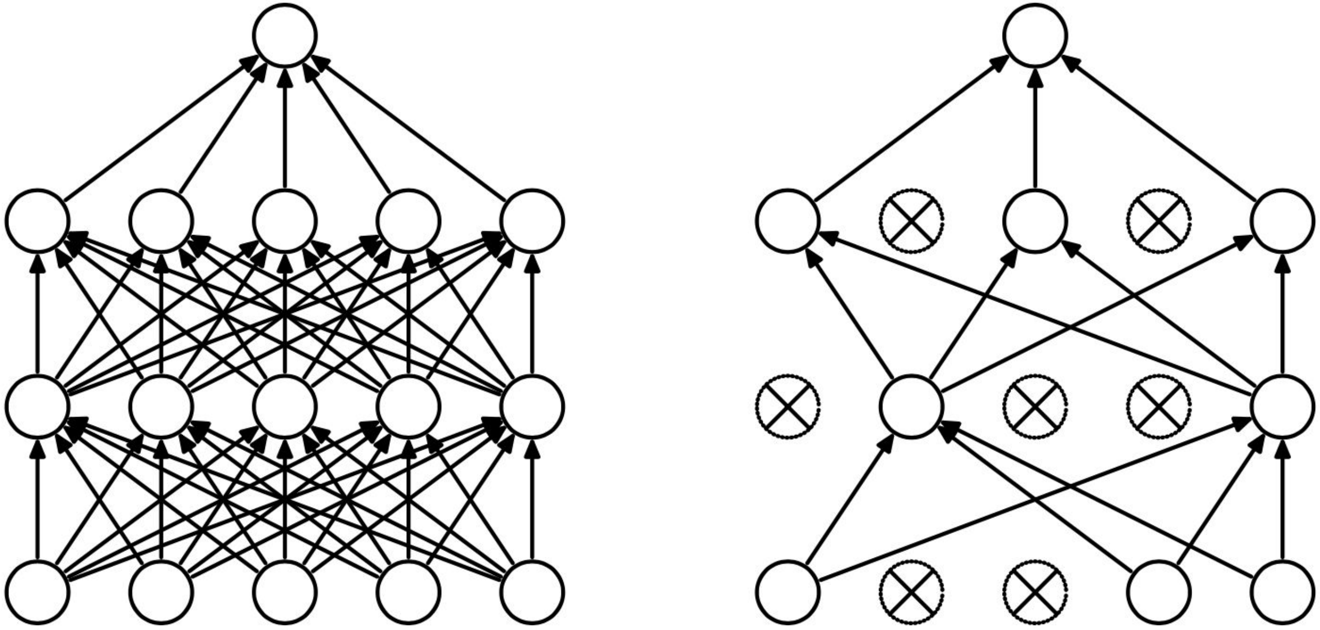


Image source: Guo+ 2017 "On calibration of modern neural networks"

Caveat: Dataset shift...

Simple Baseline: Monte Carlo Dropout



(a) Standard Neural Net (b) After applying dropout.

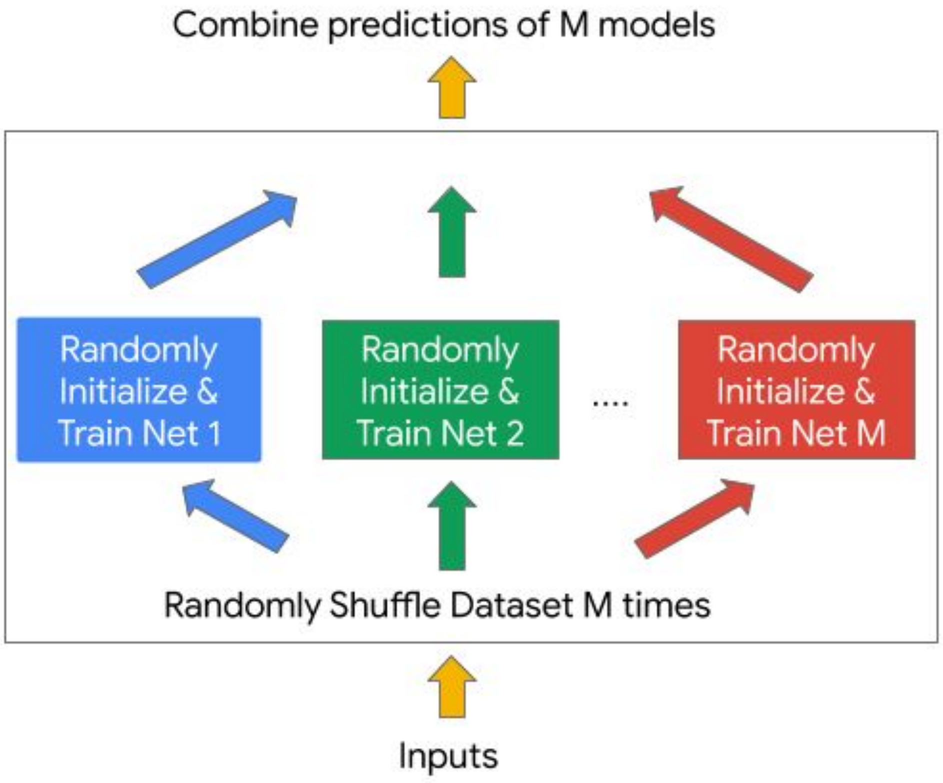
Image source: Dropout: A Simple Way to Prevent Neural Networks from Overfitting

[Gal+ 2015]

Simple Baseline: Deep Ensembles

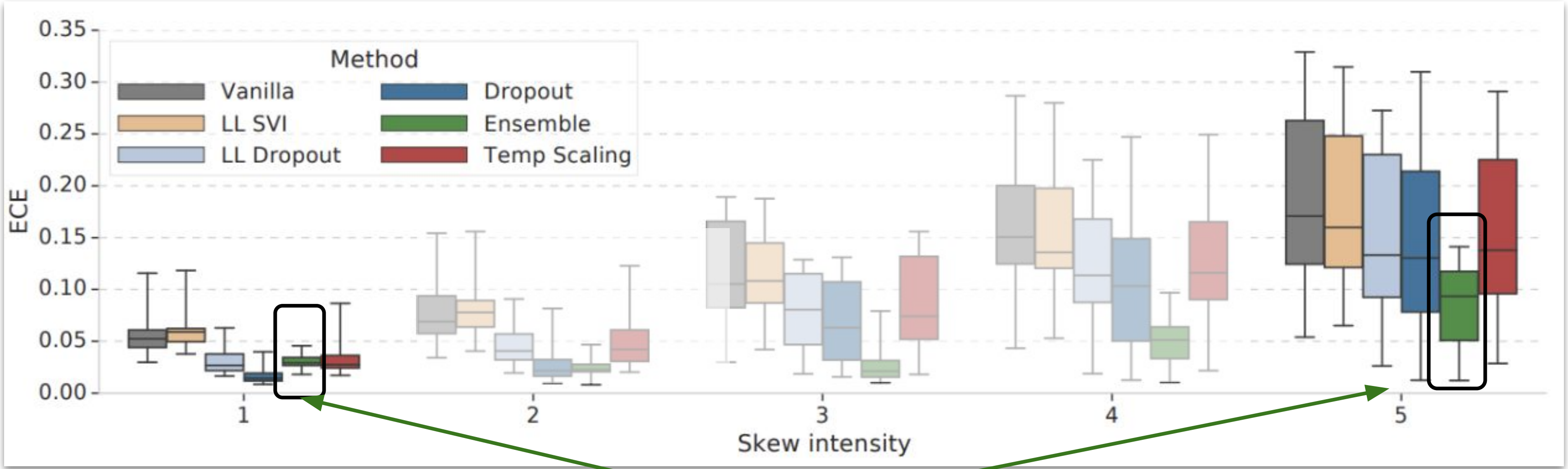
Idea: Just re-run standard SGD training but with different random seeds and average the predictions

- A well known trick for getting better accuracy and Kaggle scores
- We rely on the fact that the loss landscape is non-convex to land at different solutions
  - Rely on different initializations and SGD noise



[Lakshminarayanan+ 2017]

Deep Ensembles work surprisingly well in practice



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

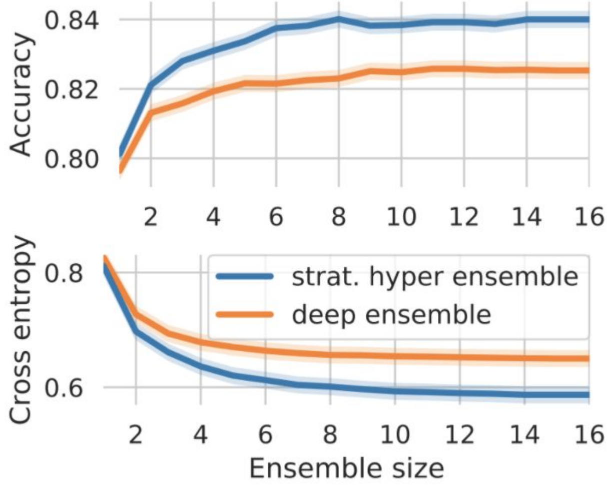
Hyperparameter Ensembles

Google AI
Brain Team

Deep ensembles differ only in random seed. By expanding the space of hyperparameters we average over, we can get even better accuracy & uncertainty estimates.

1. Run random search to generate a set of models.
   a. Include random seed as part of the search space.

2. Greedily select the K models to pool.

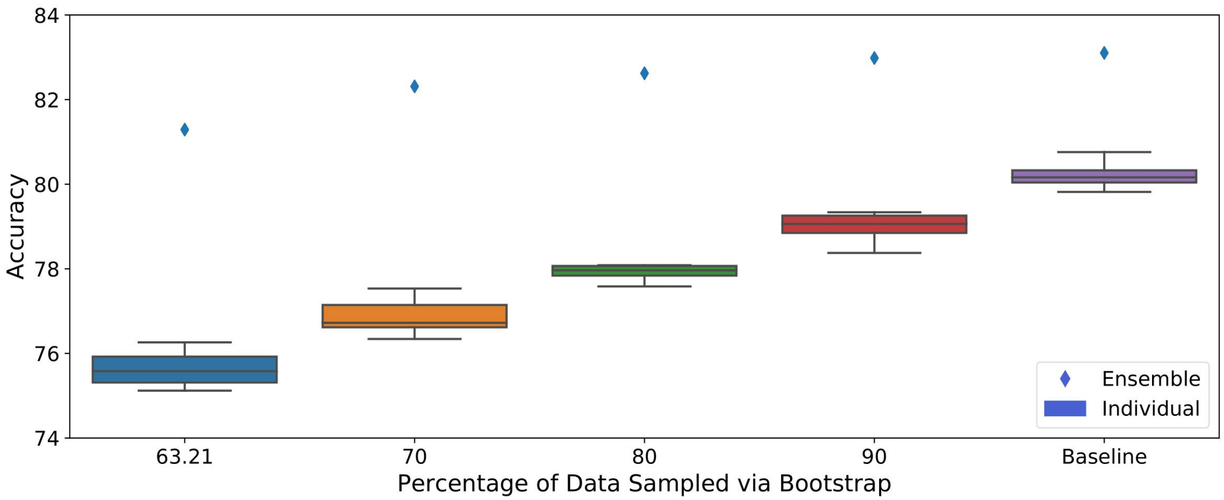


[Wenzel+ 2020 @ NeurIPS this year]

Simple Baseline: Bootstrap

Classic method for estimating uncertainty in statistical models [Efron 1979]

- Resample the dataset with replacement and retrain
- Each example gets a different weight under each model

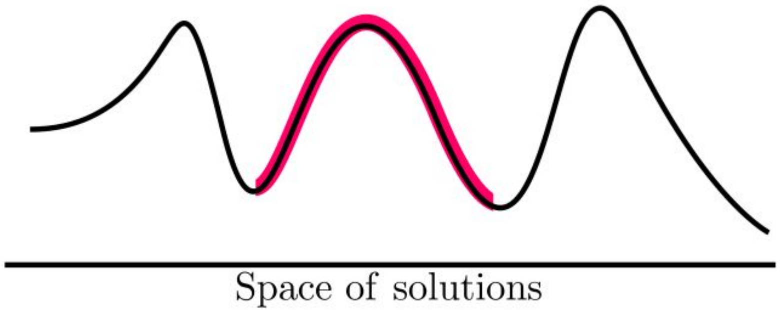


[Nixon+ 2020 @ NeurIPS workshop: I Can't Believe It's Not Better!]

Simple Baseline: SWAG + Laplace

Fit a simple distribution to the mode centered around the SGD solution

- SWAG: Fit a Gaussian around averaged weight iterates near the mode
- Laplace: Fit a quadratic at the mode, using the Hessian or Fisher information



[Maddox+ 2019]

Coffee Break (15 mins)

Check out the Q&A.

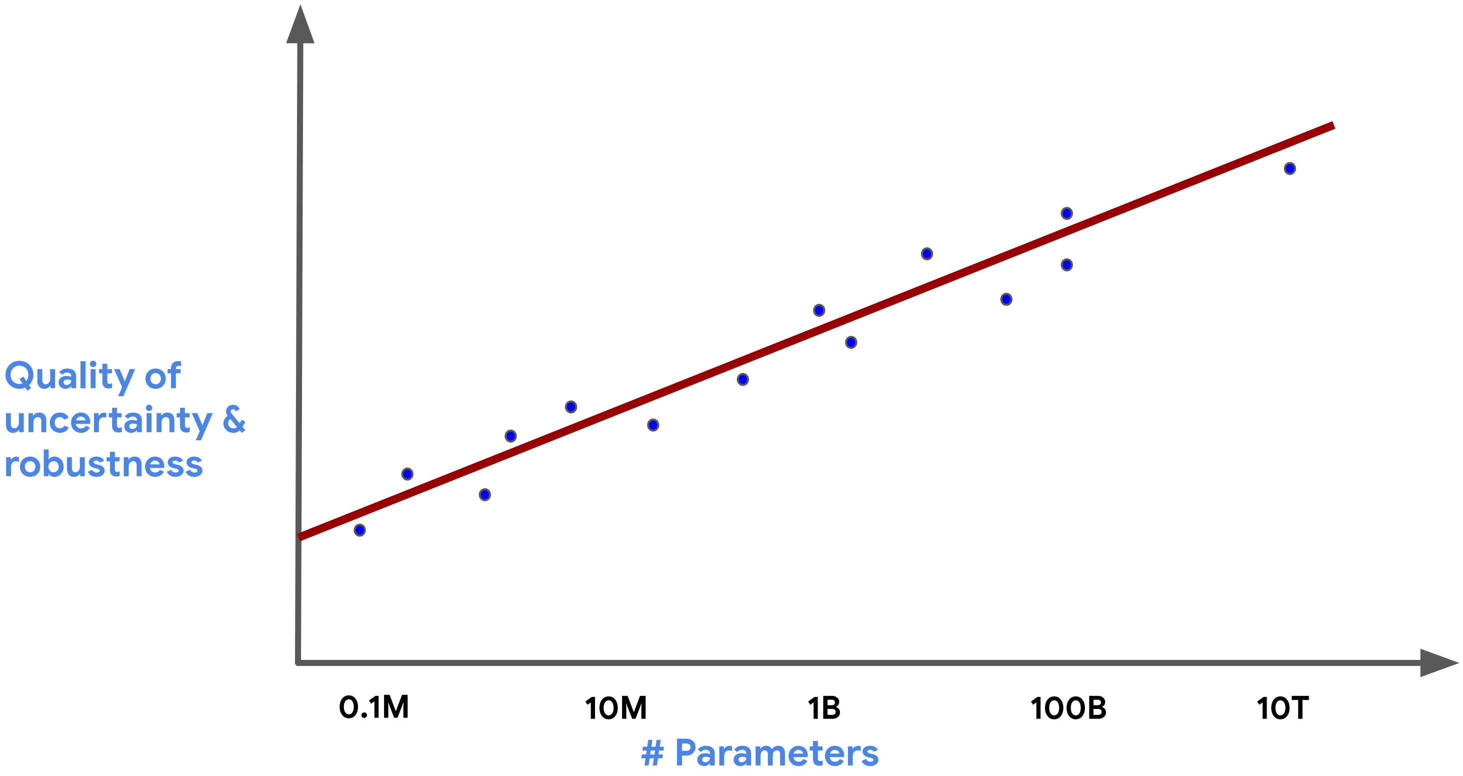
1. Why uncertainty & robustness?

2. Foundations.

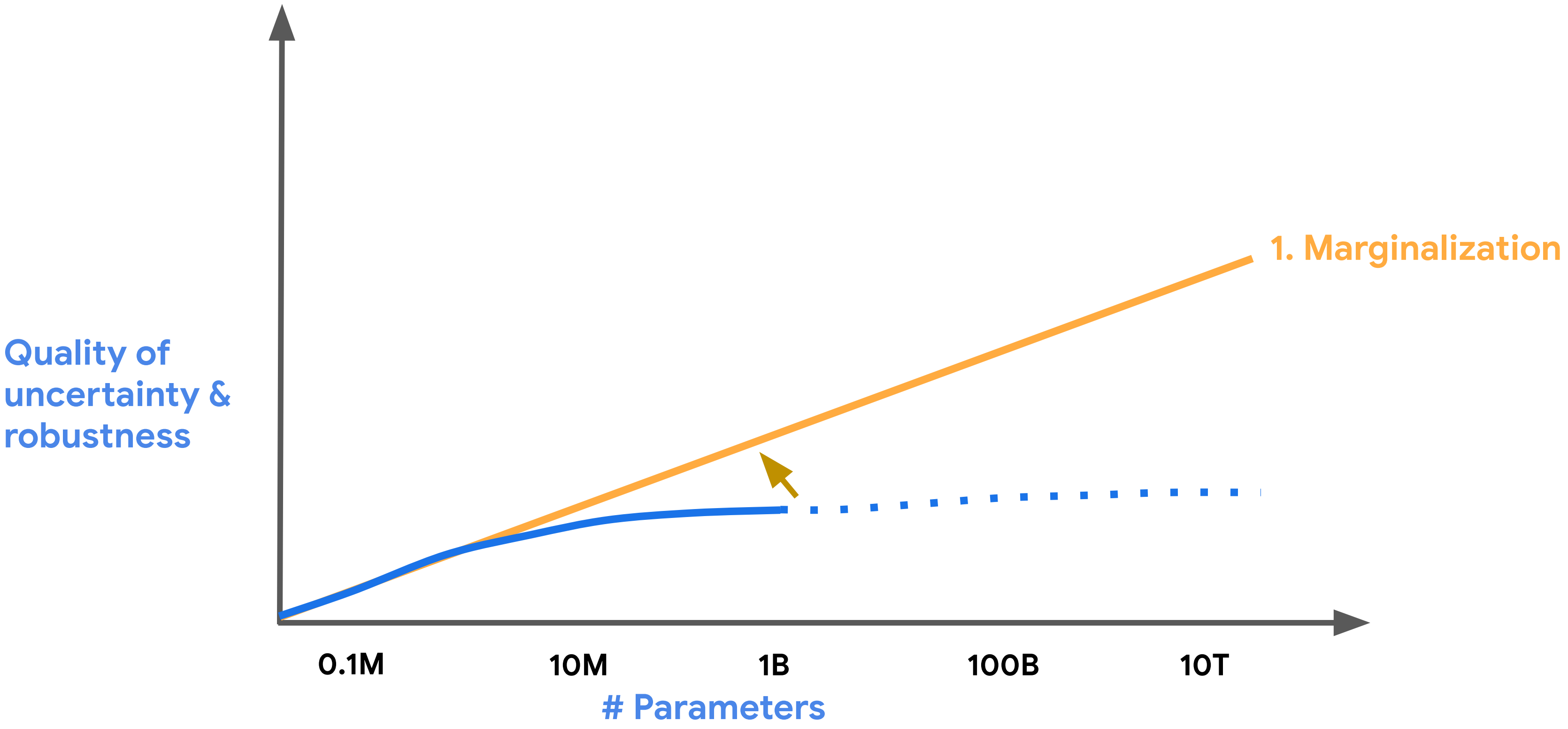
3. Recent advances.

What about scale?

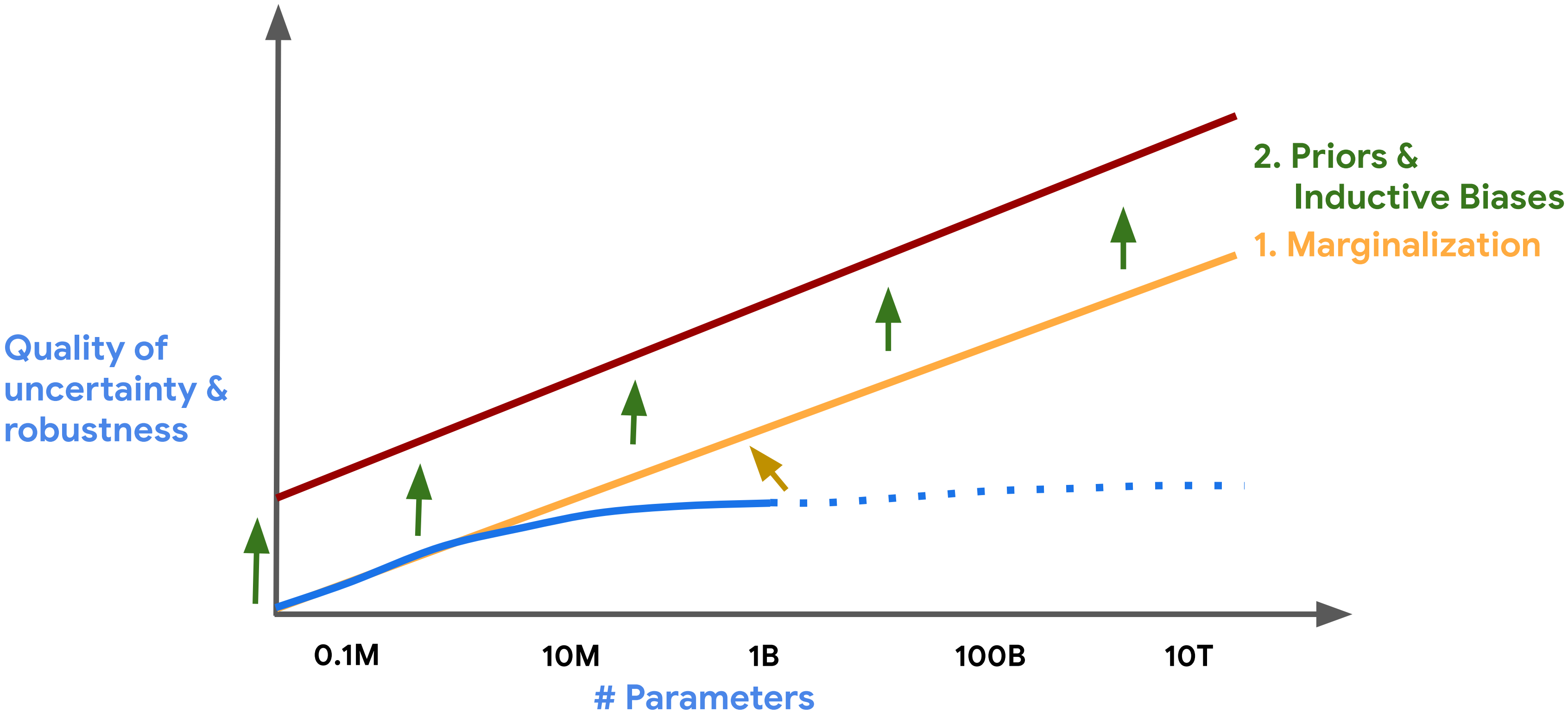
The **uncertainty-robustness frontier**



The **uncertainty-robustness frontier**



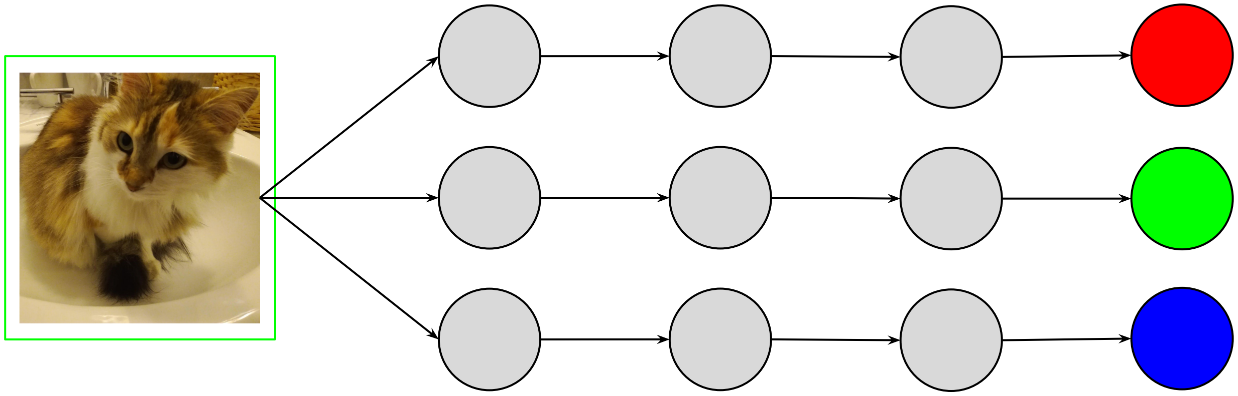
The **uncertainty-robustness frontier**



Marginalization

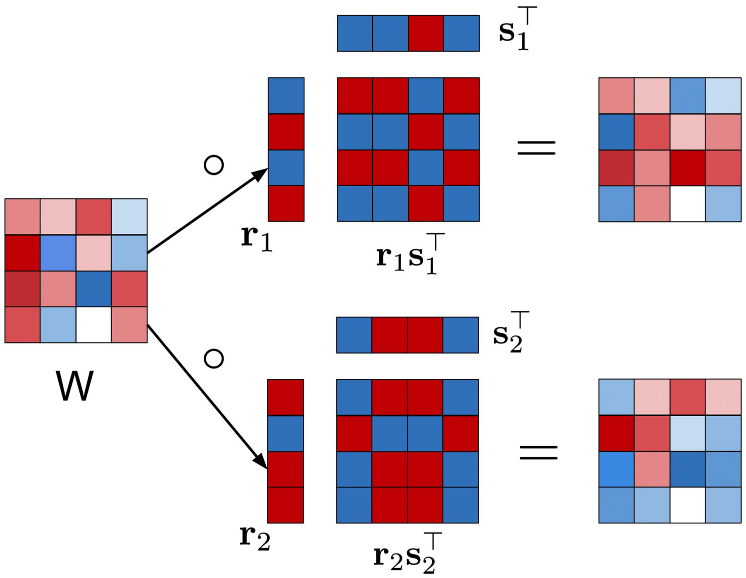
Ensembles as a Giant Model

We can trace the frontier by providing a perspective of ensembles as a single model.



- Paths between subnetworks are independent ⇒ SGD-trained models have independent predictions by construction.
- Bridge the gap from single model to ensembles by *sharing parameters*, learning how to decorrelate predictions *during training*.

Efficient Ensembles by Sharing Parameters



Parameterize each weight matrix as a new weight matrix ***W*** multiplied by the outer product of two vectors ***r*** and **s**.

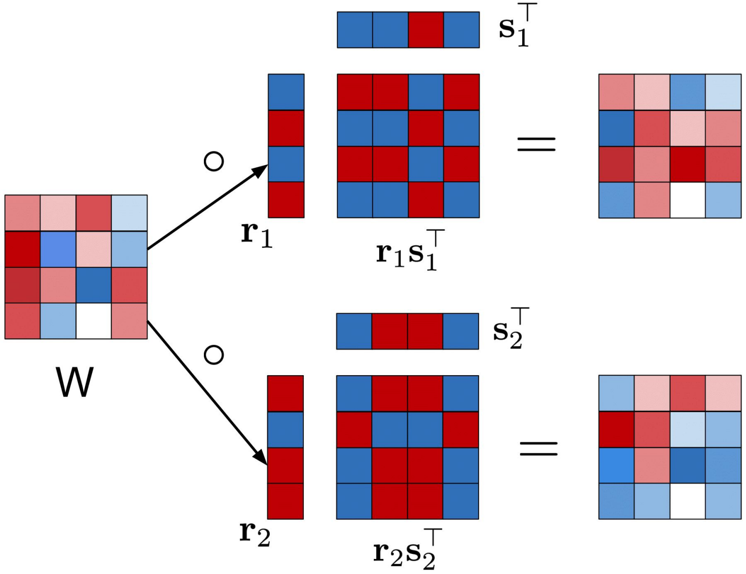
$$\overline{W}_i = W \circ F_i, \text{ where } F_i = s_i r_i^\top$$

There is an independent set of ***r*** and **s** vectors for each ensemble member; ***W*** is shared.

Known as **BatchEnsemble**.

[Wen+ 2020]

Efficient Ensembles by Sharing Parameters



BatchEnsemble has a convenient vectorization.

Duplicate each example in a given mini-batch *K* times.

$$Y = \phi(((X \circ S)W) \circ R)$$

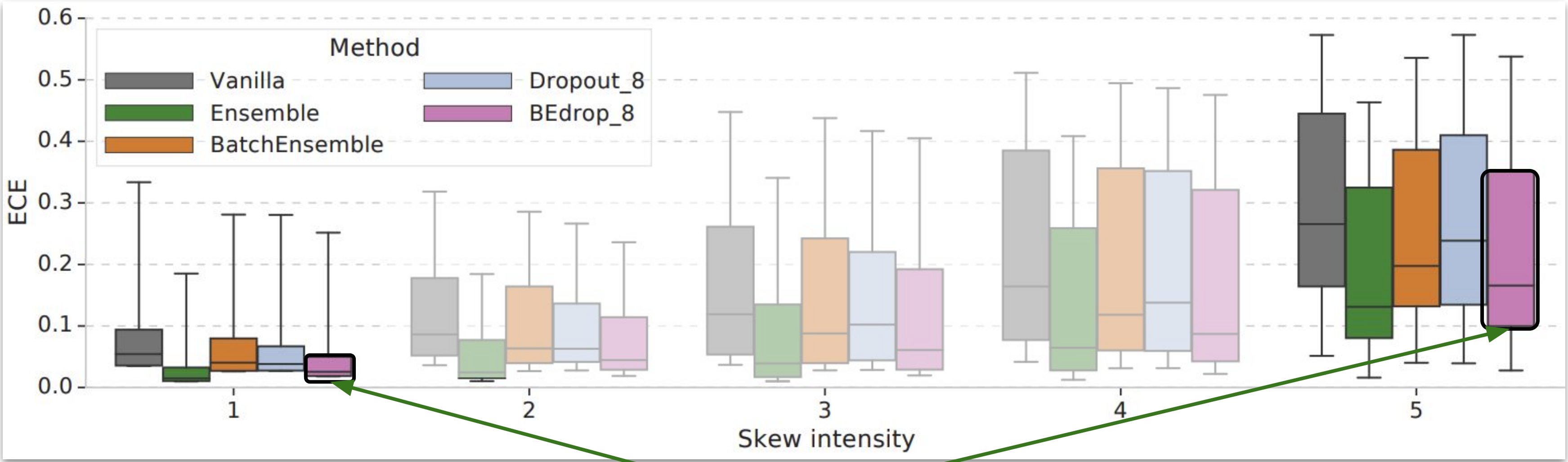
The model yields *K* outputs for each example.

Can interpret rank-1 weight perturbations as *feature-wise transformations*.

[Wen+ 2020]

Google AI
Brain Team

BatchEnsemble works surprisingly well in practice

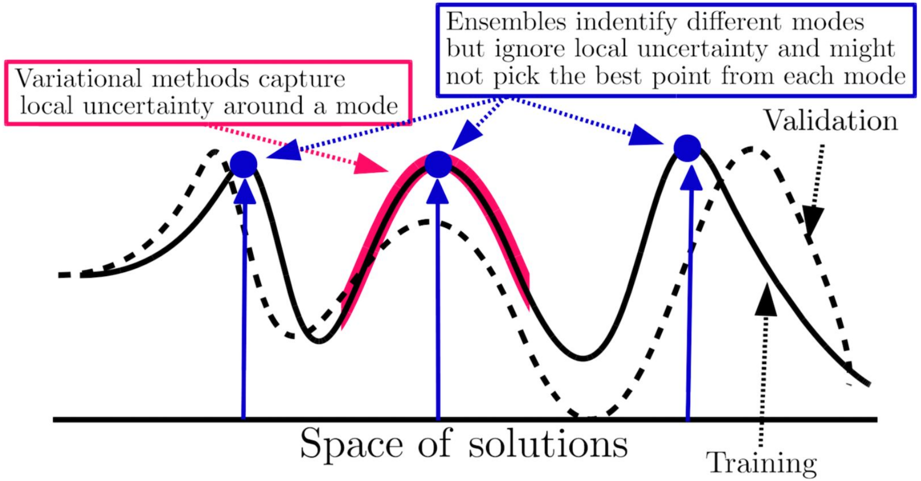


BatchEnsemble is consistently the best performing method given # parameters.

What happened to Bayesian neural nets?

Google AI
Brain Team

By analyzing loss surfaces, can show that Variational Bayesian neural nets are effective at **averaging uncertainty within a single mode**. They **fail to explore the full space.**



Can we further ensembles with ideas from Bayesian neural nets?

[Fort+ 2019]

Rank-1 Bayesian Neural Networks

Google AI
Brain Team

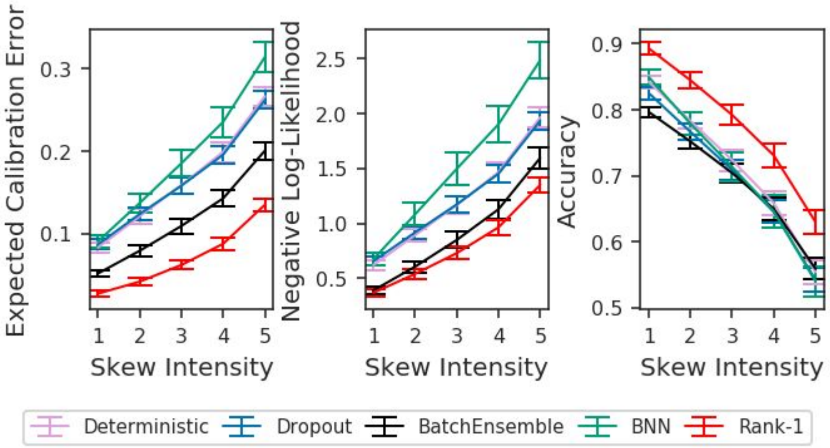
1. Start from BatchEnsemble's parameterization.

2. Add priors over rank-1 weights p(**r**), p(**s**).

$$p(\mathbf{W}') = \iint \mathcal{N}(\mathbf{W}' \mid 0, (\mathbf{r}\mathbf{s}^T\sigma)^2) p(\mathbf{r}) p(\mathbf{s}) \, \mathbf{d}\mathbf{r} \, \mathbf{d}\mathbf{s}$$

3. Use mixture variational posteriors.

$$q(\mathbf{r}) = \frac{1}{K} \sum \pi_k q(\mathbf{r}_k; \boldsymbol{\lambda}_k)$$



Rank-1 BNNs combine **local** and **global** behavior.

See also cyclical MCMC [Zhang+ 2020].

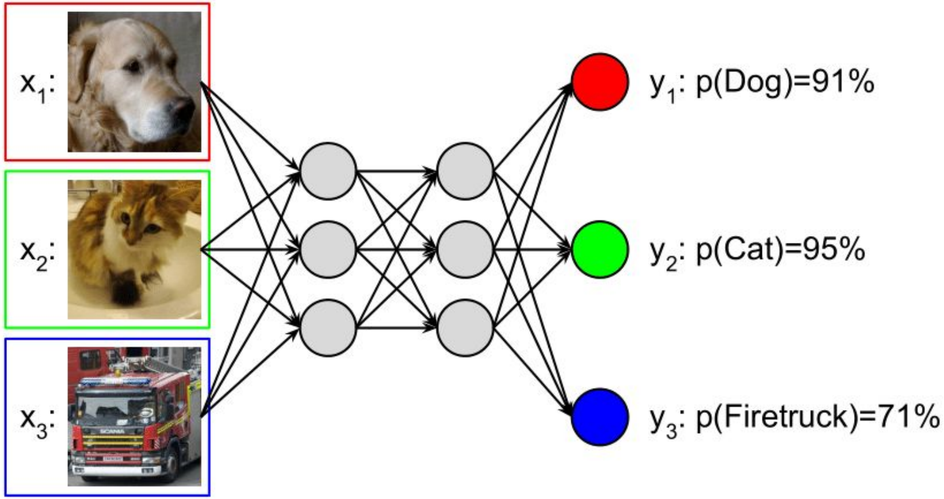
[Dusenberry+ 2020]

Toward simpler & faster models

Google AI
Brain Team

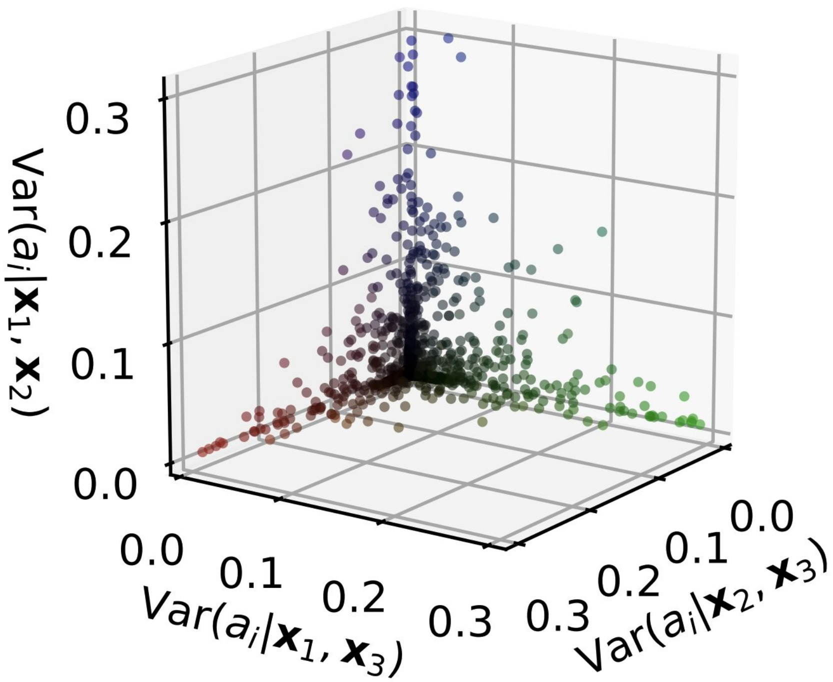
Recently, we found you can get the same results with an even simpler configuration: multi-input multi-output (MIMO).

Instead of low-rank perturbations, rely on subnetwork paths learned implicitly during training.



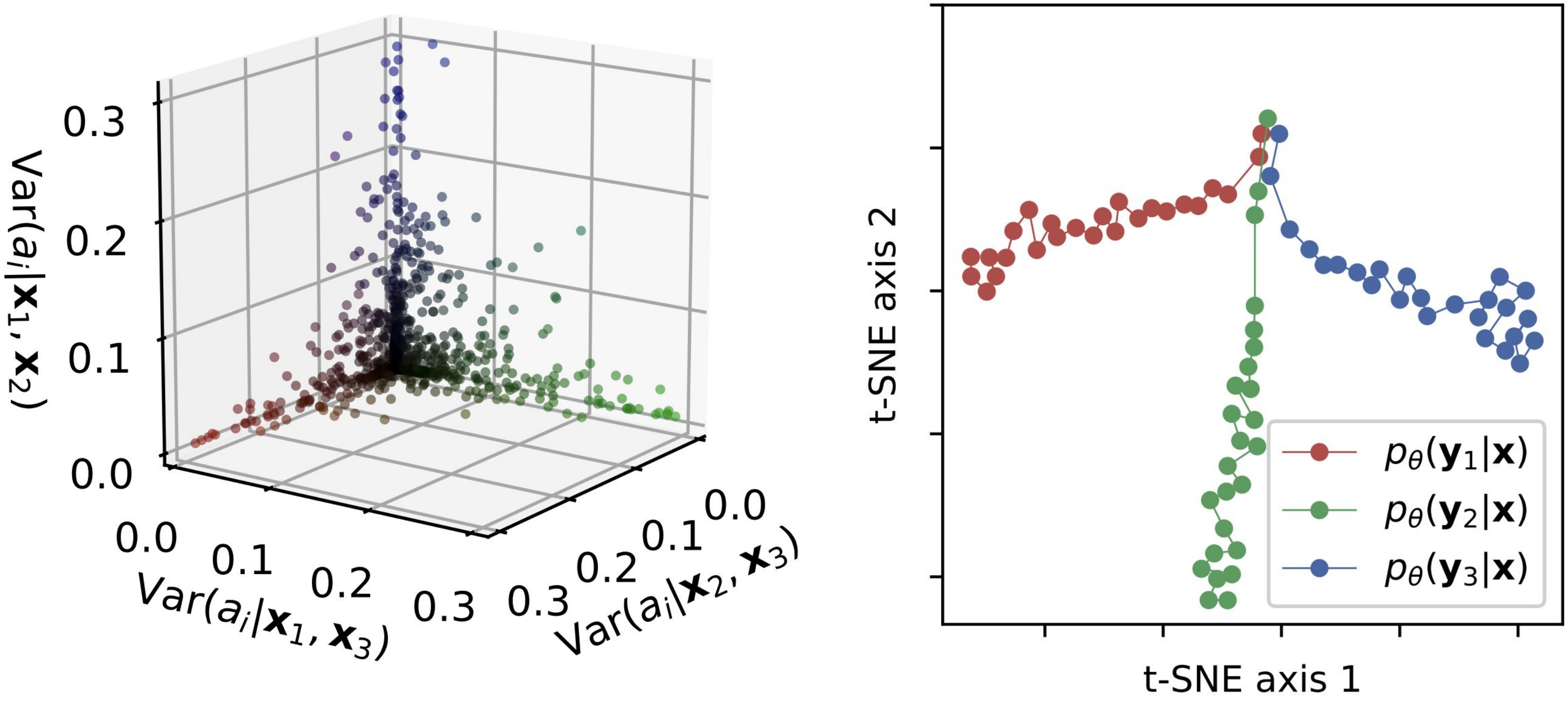
[Havasi+ 2020]

Toward simpler & faster models



[Havasi+ 2020]

Toward simpler & faster models



[Havasi+ 2020]

Priors & Inductive Biases

How do we select the prior?

Standard normal prior N(0, 1) is the default. But.. it’s not great.

How do we select the prior?

Standard normal prior N(0, 1) is the default. But.. it's not great.

- It has bad statistical properties.
    - Does not leverage information about the network structure.
    - In the limit, all hidden units contribute infinitesimally to each input. [Neal 1994]
    - Unclear how to encourage predictive behavior, e.g., robustness to specific OOD.

How do we select the prior?

Standard normal prior N(0, 1) is the default. But.. it's not great.

- It has bad statistical properties.
  - Does not leverage information about the network structure.
  - In the limit, all hidden units contribute infinitesimally to each input. [Neal 1994]
  - Unclear how to encourage predictive behavior, e.g., robustness to specific OOD.
- It has bad optimization properties.
  - Sensitive to parameterization.
  - Too strong a regularizer. [Bowman+ 2015; Trippe Turner 2018]

Google AI Brain Team

How do we select the prior?

Standard normal prior N(0, 1) is the default. But.. it's not great.

- It has bad statistical properties.
  - Does not leverage information about the network structure.
  - In the limit, all hidden units contribute infinitesimally to each input. [Neal 1994]
  - Unclear how to encourage predictive behavior, e.g., robustness to specific OOD.
- It has bad optimization properties.
  - Sensitive to parameterization.
  - Too strong a regularizer. [Bowman+ 2015; Trippe Turner 2018]

We often have more intuition in terms of input-output relationships: function priors. [Hafner+ 2018, Sun+ 2019]

Priors can be non-probabilistic, coming in the form of structural biases.

Inductive biases can arise from architecture considerations.

What about inductive biases to assist OOD?

- Hypothesis: “*Representations should be invariant with respect to dataset shift.*”
- **Data augmentation** extends the dataset in order to encourage invariances.
- More examples: **contrastive learning**, **equivariant architectures**.

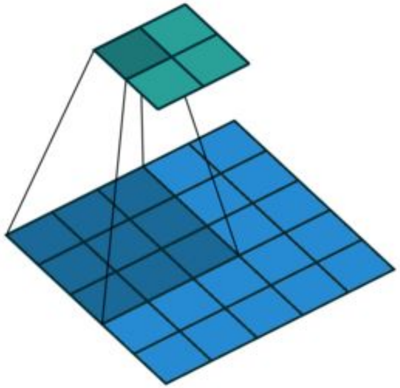


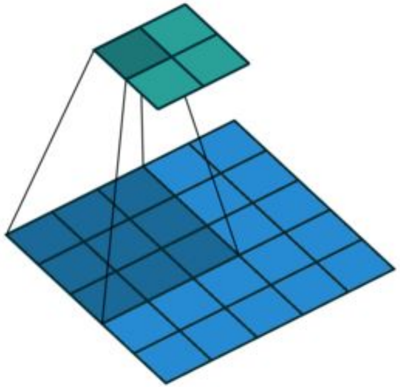
Image source: Dumoulin & Visin 2016

Priors can be non-probabilistic, coming in the form of structural biases.

Data augmentation requires two considerations:

1. Set of base augmentation operations.
   (Ex: color distortions, word substitution)

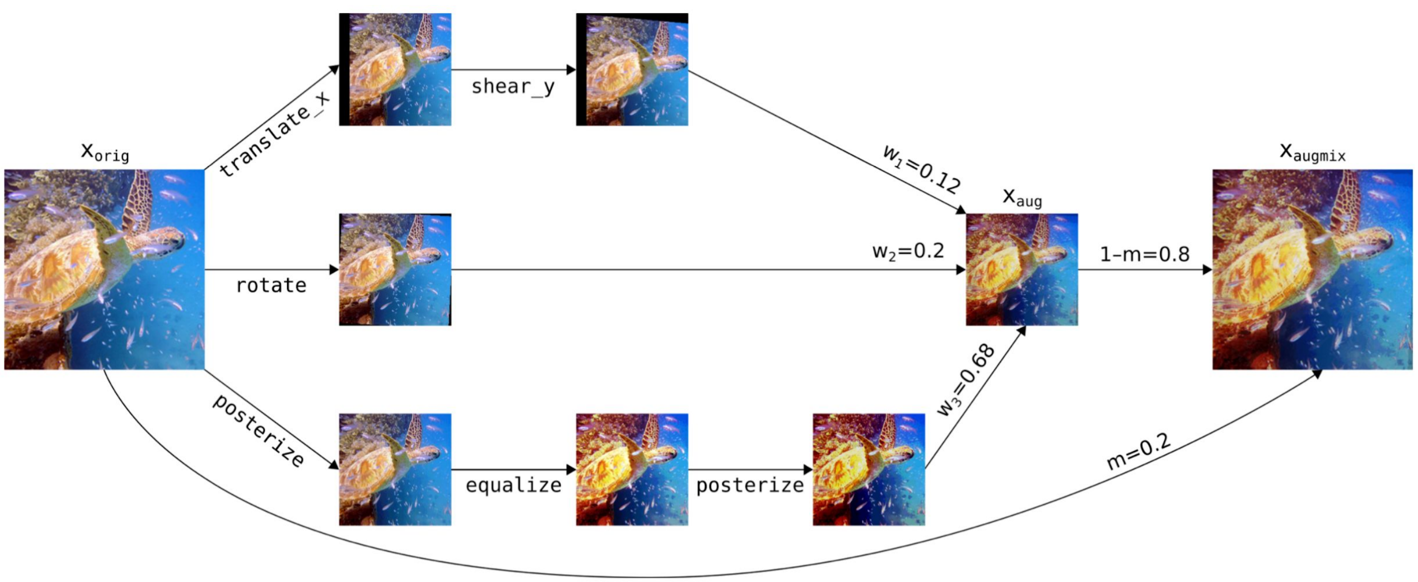
2. Combination strategy.
   (Ex: Sequence of K randomly selected ops.)



See [Hafner+ 2018] for a probabilistic interpretation.

Image source: Dumoulin & Visin 2016

Composing a set of base augmentations

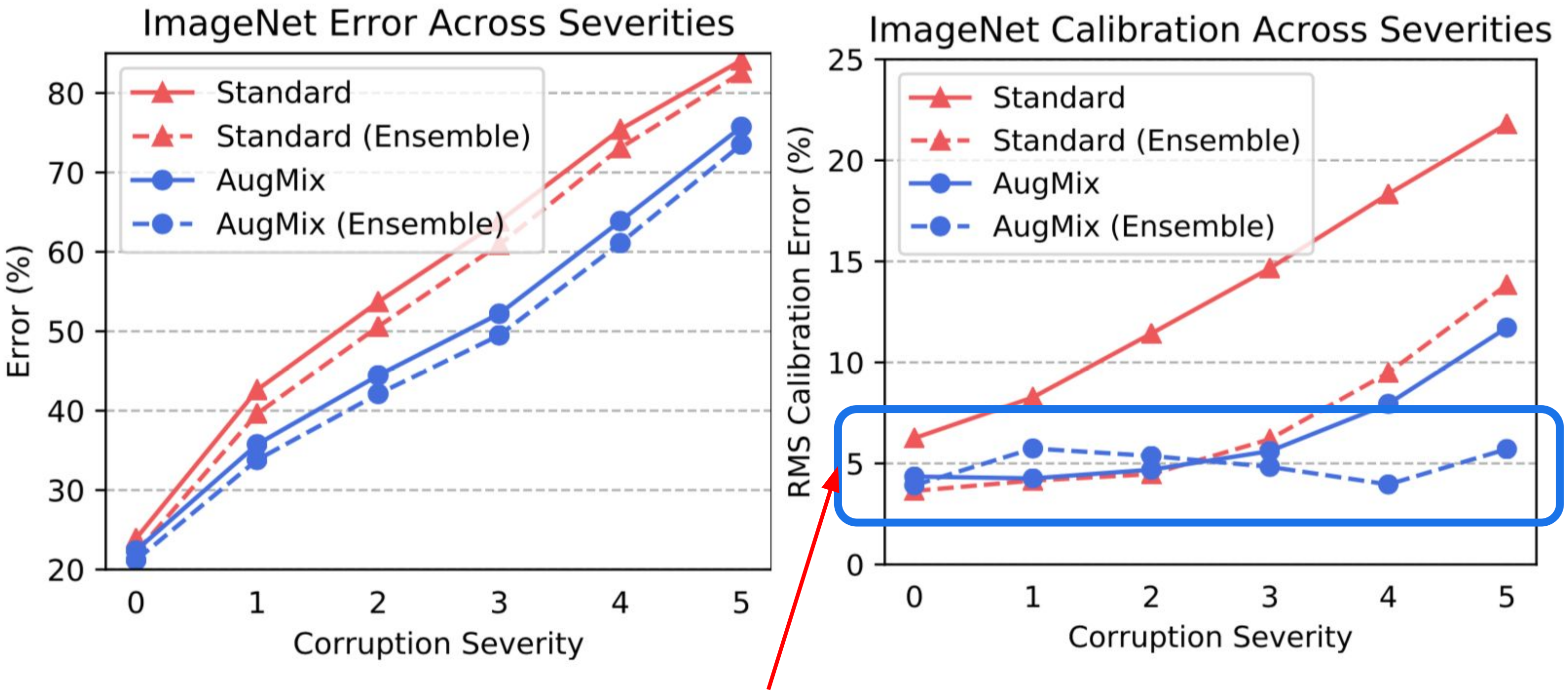


Composing base operations and ‘mixing’ them can improve accuracy and calibration under shift.

[Hendrycks+ 2020]

AugMix improves accuracy & calibration under shift

Google AI Brain Team



Data augmentation can provide complementary benefits to marginalization.

[Hendrycks+ 2020]

Imposing distance awareness

Google AI
Brain Team

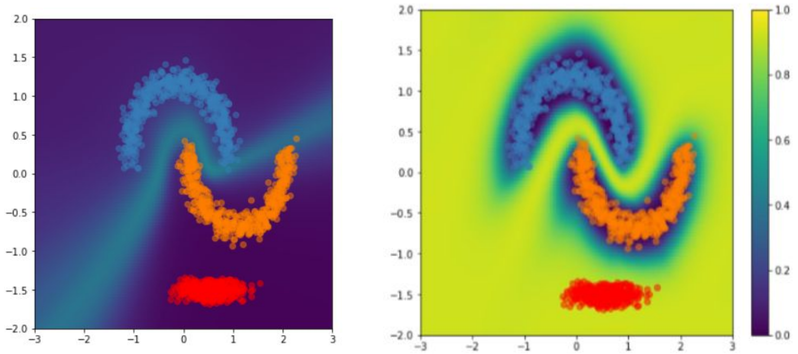
Data augmentation is effective for enforcing invariant predictions under shift.

"Models should be distance aware: uncertainty increases farther from training data."

Spectral-normalized Neural Gaussian process

1. Replace output layer with "GP layer".
2. Apply spectral normalization to preserve input distances within internal layers.

See also [van Amersfoort+ 2020].



Deep Ensemble SNGP

[Liu+ 2020 @ NeurIPS this year]

Imposing distance awareness

Google AI Brain Team

Data augmentation is effective for enforcing invariant predictions under shift.

“Models should be distance aware: uncertainty increases farther from training data.”

Spectral-normalized Neural Gaussian process

1. Replace output layer with “GP layer”.
2. Apply spectral normalization to preserve input distances within internal layers.

See also [van Amersfoort+ 2020].

BERT on an intent detection benchmark

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

[Liu+ 2020 @ NeurIPS this year]

Wrapping Up

Google AI
Brain Team

Open Challenge: Scale

Enable uncertainty & robustness at the billion-trillion parameter scale.

Datasets. What is the role of priors on increasingly larger and diverse datasets?

Tasks. How do we think about OOD as we move toward general solutions to a wide range of tasks?

Model Parallelism. Mixtures of experts are already the backbone. Can we exploit recent ideas to enable even bigger and adaptive models?

Google AI Brain Team

Open Challenge: Understanding

Why are the best models the best? How do we close the gap from theory to practice?

One promising perspective is generalization theory in deep learning. PAC-Bayes provides explicit bounds on the generalization error of neural networks.

- In benchmarks, PAC-Bayes measures correlate best with generalization. [Jiang+ 2020]
- There are exact ties between ensemble diversity and tighter generalization bounds. [Masegosa 2020]

Check out [NeurIPS workshop: I Can't Believe It's Not Better!]

Open Challenge: Benchmarks

In the past few years, there's been an ongoing call to action on benchmarks:

- *Comprehensive baselines* across standard and SOTA methods.
- *Large-scale models & datasets.*
- *High-quality code*: small changes in the setup can dramatically affect performance.

Preliminary efforts exist but a unified effort is required.

Today, we're happy to announce we made progress on this challenge.

Uncertainty Baselines

github.com/google/uncertainty-baselines

High-quality implementations of baselines on a variety of tasks.

Ready for use: 7 settings, including:

- Wide ResNet 28-10 on CIFAR
- ResNet-50 and EfficientNet on ImageNet
- BERT on Clinc Intent Detection

14 different baseline methods.

Used across **10** projects at Google.

Collaboration with OATML @ Oxford, unifying github.com/oatml/bdl-benchmarks.

dustinvtran and edward-bot Retune VI baseline for CIFAR. ... Latest commit 9379550 3 hours ago

..

README.md	Retune VI baseline for CIFAR.	3 hours ago
batchensemble.py	Move baselines/cifar10/ to baselines/cifar/.	13 days ago
batchensemble_model.py	Move baselines/cifar10/ to baselines/cifar/.	13 days ago
batchensemble_model_test.py	Move baselines/cifar10/ to baselines/cifar/.	13 days ago
deterministic.py	Move baselines/cifar10/ to baselines/cifar/.	13 days ago
deterministic_test.py	Move baselines/cifar10/ to baselines/cifar/.	13 days ago
dropout.py	Move baselines/cifar10/ to baselines/cifar/.	13 days ago
dropout_test.py	Move baselines/cifar10/ to baselines/cifar/.	13 days ago
ensemble.py	Move baselines/cifar10/ to baselines/cifar/.	13 days ago
utils.py	Move baselines/cifar10/ to baselines/cifar/.	13 days ago
variational_inference.py	Retune VI baseline for CIFAR.	3 hours ago
variational_inference_test.py	Move baselines/cifar10/ to baselines/cifar/.	13 days ago

README.md

Wide ResNet 28-10 on CIFAR

CIFAR-10

Method	Train/Test NLL	Train/Test Accuracy	Train/Test Cal. Error	cNLL/cA/cCE	Train Runtime (hours)	# Parameters
Deterministic	1e-3 / 0.159	99.9% / 96.0%	1e-3 / 0.0231	1.29 / 69.8% / 0.173	1.2 (8 TPUv2 cores)	36.5M
BatchEnsemble (size=4)	0.08 / 0.143	99.9% / 96.2%	5e-5 / 0.0206	1.24 / 69.4% / 0.143	5.4 (8 TPUv2 cores)	36.6M
Dropout	2e-3 / 0.160	99.9% / 95.9%	2e-3 / 0.0241	1.35 / 67.8% / 0.178	1.2 (8 TPUv2 cores)	36.5M
Ensemble (size=4)	2e-3 / 0.114	99.9% / 96.6%	-	-	1.2 (32 TPUv2 cores)	146M
Variational inference	1e-3 / 0.211	99.9% / 94.7%	1e-3 / 0.029	1.46 / 71.3% / 0.181	5.5 (8 TPUv2 cores)	73M

Robustness Metrics

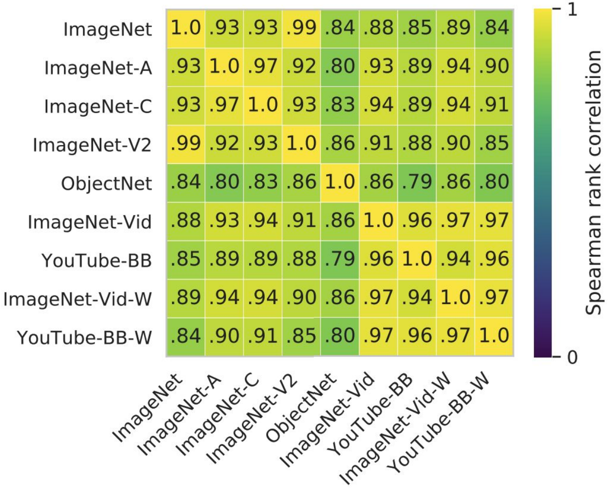
github.com/google-research/robustness_metrics

Lightweight modules to evaluate a model's robustness and uncertainty predictions.

Ready for use:

- 10 OOD datasets
- Accuracy, uncertainty, and stability metrics
- Many SOTA models (TFHub support!)
- Multiple frameworks (JAX support!)

Enables large-scale studies of robustness [Djolonga+ 2020].



Collaboration lead by Google Research, Brain Team @ Zurich.

Takeaways

- Uncertainty & robustness are critical problems in AI and machine learning.
- Benchmark models with calibration error and a large collection of OOD shifts.
- Probabilistic ML, ensemble learning, and optimization provide a foundation.
- The best methods advance two dimensions: combining multiple neural network predictions; and imposing priors and inductive biases.
- As compute increases, the uncertainty-robustness frontier outlines future progress.

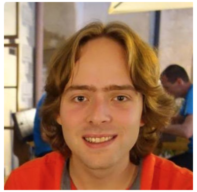
Thank you!

Google AI
Brain Team


Ben Adlam


Josip Djolonga


Mike Dusenberry

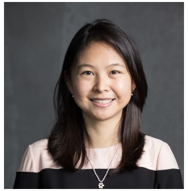

Stanislav Fort


Justin Gilmer


Marton Havasi

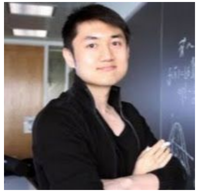

Danijar Hafner


Dan Hendrycks


Clara Hu


Rodolphe Jenatton


Ghassen Jerfel


Jeremiah Liu


Mario Lucic


Zelda Mariet


Rafael Müller


Kevin Murphy

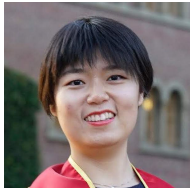

Zack Nado


Eric Nalisnick


Kathleen Nix


Jeremy Nixon


Shreyas Padhy


Jie Ren


D. Sculley


Yeming Wen


Florian Wenzel

& others!