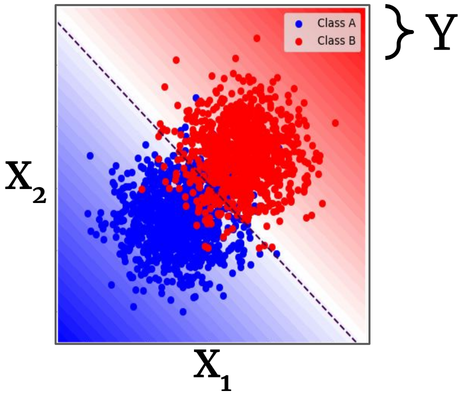
Uncertainty in Deep Learning

Balaji Lakshminarayanan, Dustin Tran, Jasper Snoek

Background

What do we mean by Predictive Uncertainty?

- Predict output distribution p(y|x) rather than point estimate, e.g.
  - ○ ***Classification***: output label along with confidence
  - ○ ***Regression***: output mean and variance



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

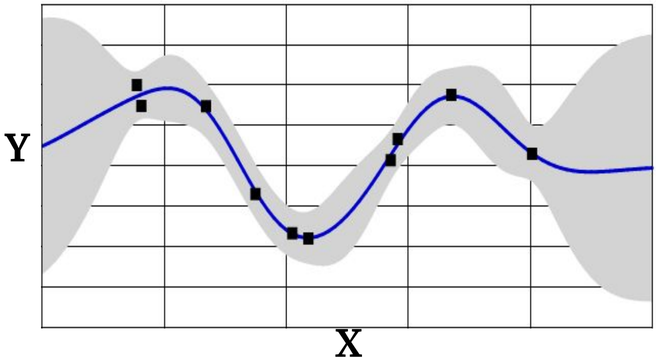


Image credit: Eric Nalisnick

Sources of uncertainty: *Inherent ambiguity*

Google AI
Brain Team

- Noise in the labeling process (humans disagree on the label, e.g. CIFAR-10-H)
- Measurement noise in y
- Also known as ***aleatoric uncertainty***
- Considered to be "***irreducible uncertainty***"
  - Persists even in the limit of infinite data
  - Partial observability: could be reduced given additional features

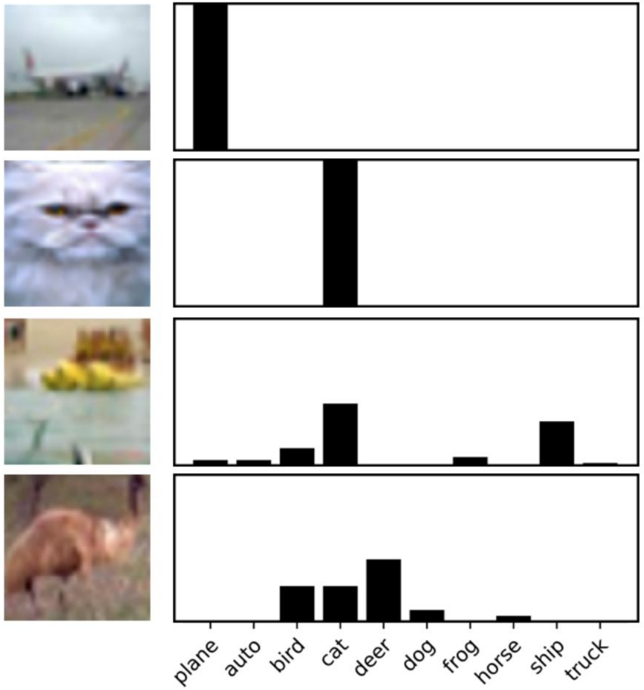
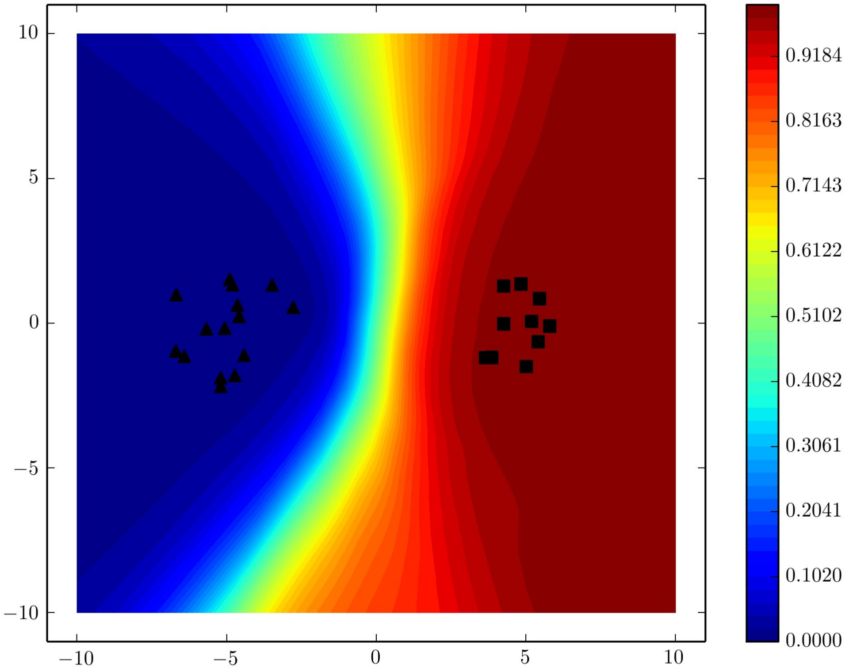


Image source: Battleday et al. 2019 "Improving machine classification using human uncertainty measurements"

Sources of uncertainty: *Model uncertainty*

- Multiple parameters could be consistent with the observed training data
- Also known as ***epistemic uncertainty***
- Considered to be "**reducible uncertainty**"
    - Vanishes in the limit of infinite data (subject to model identifiability)



How do we measure the quality of uncertainty?

Calibration measures how well predicted confidence (probability of correctness) aligns with the observed accuracy.

- Expected Calibration Error (ECE)
- Computed as the average gap between within-bucket accuracy and within-bucket predicted probability for S buckets.
- Does not reflect “refinement” (predicting class frequencies gives perfect calibration).

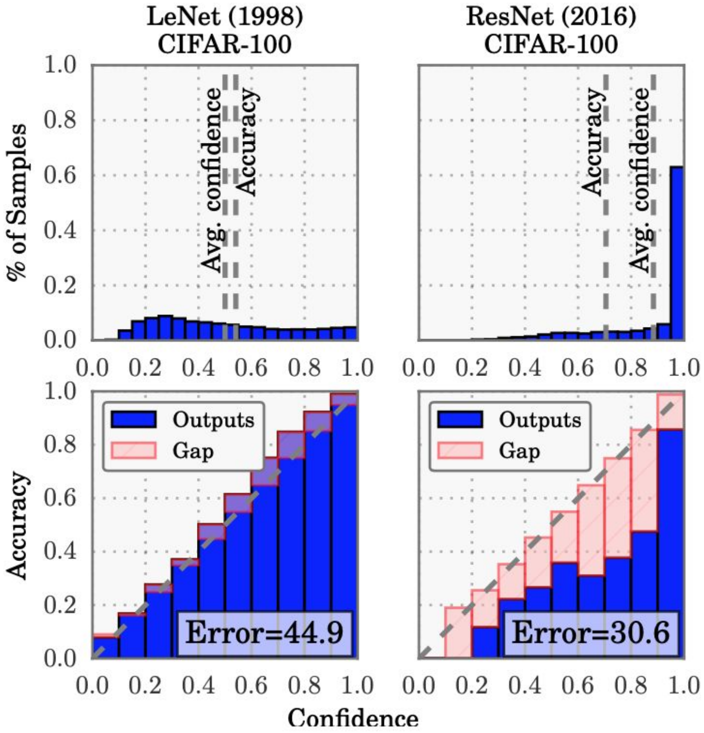


Image source: Guo et al. 2017 “On calibration of modern neural networks”

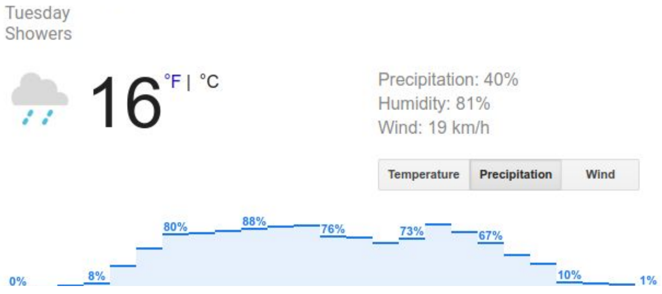
How do we measure the quality of uncertainty?

Proper scoring rules *(Gneiting & Raftery, JASA 2007),*

- Negative Log-Likelihood (NLL)
  - Can overemphasize tail probabilities

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

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



Tuesday
Showers

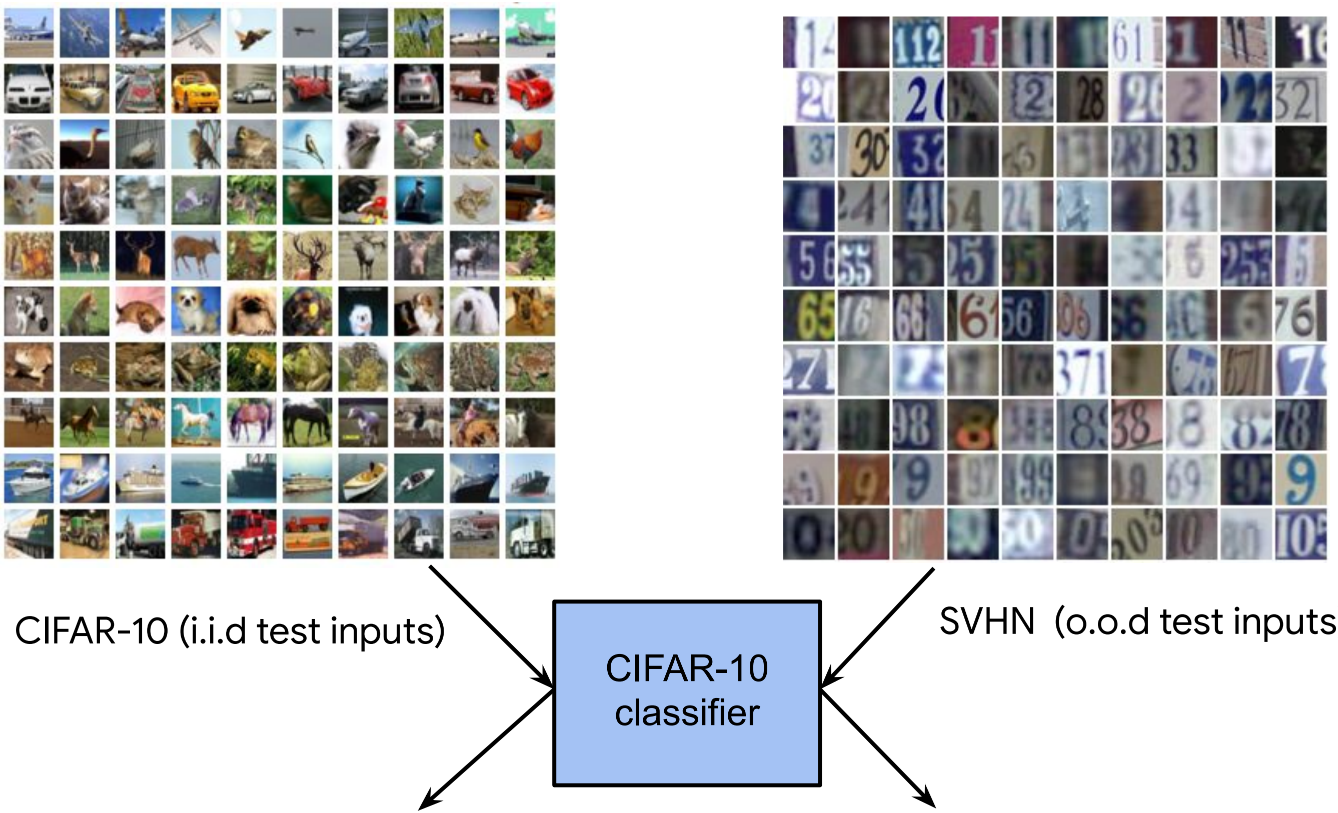
16 °F | °C

Precipitation: 40%
Humidity: 81%
Wind: 19 km/h

How do we measure the quality of uncertainty?

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 **?**

Motivating Applications

Why predictive uncertainty?

Uncertainty estimation is useful for:

- Knowing when to trust model's predictions, especially under dataset shift
- Better decision making: Calculating the risk vs reward associated with prediction (worst case vs average case)
- Active learning: Getting more data in regions where the model is uncertain
- Open set recognition
- Lifelong learning
- Exploration in Reinforcement Learning
- ...

Natural distribution shift

Dataset shift across

- Time
- Countries

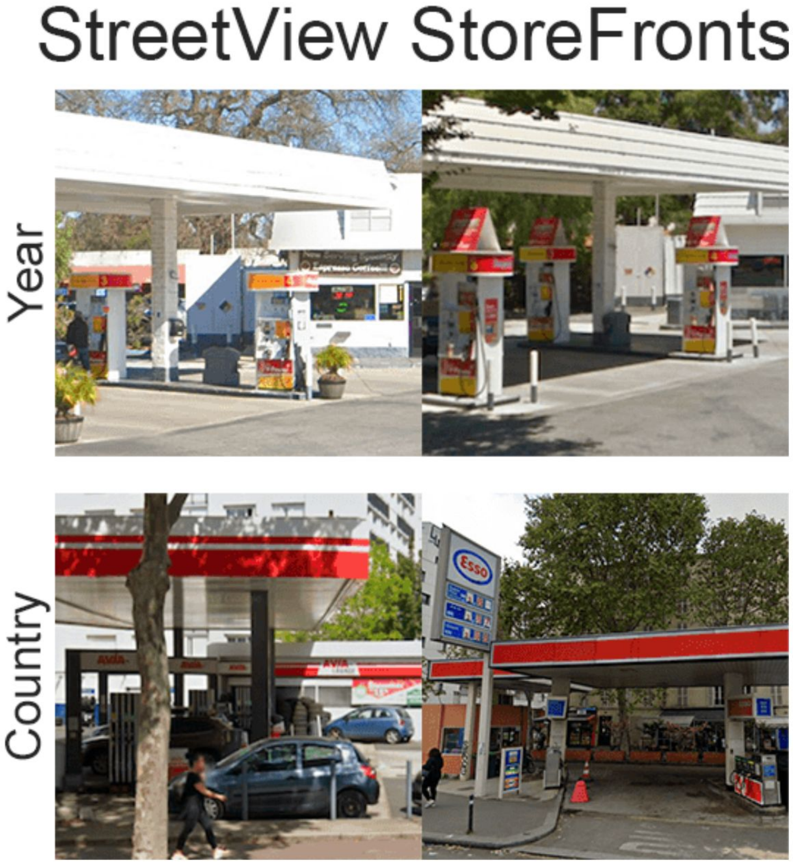


Image source: Hendrycks et al. 2020 “The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization”

Open Set Recognition

- Test inputs may not belong to one of the existing training classes
- Example: genome classifier trained on species known until time t
- Need to be able to reject such inputs as “none of the above”

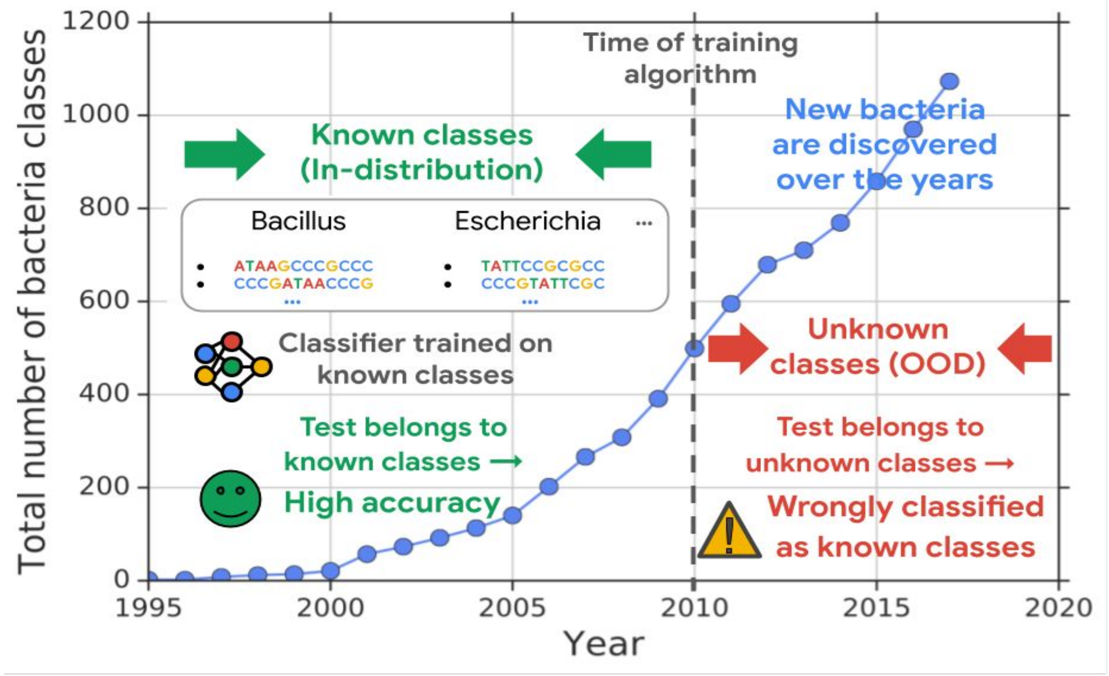
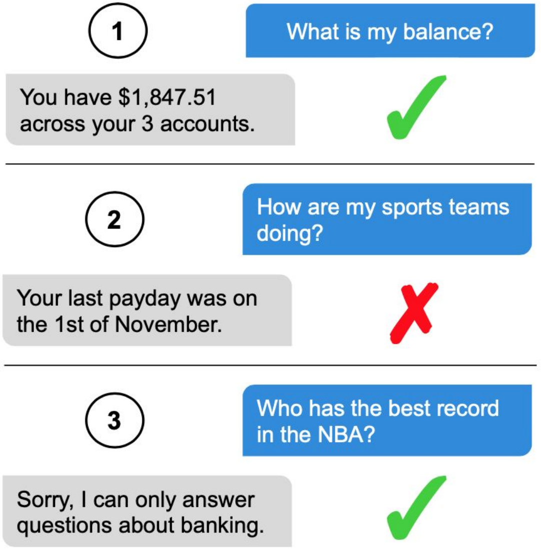


Image source: https://ai.googleblog.com/2019/12/improving-out-of-distribution-detection.html

Conversational Dialog systems

- Detecting out-of-scope utterances



1

What is my balance?

You have $1,847.51 across your 3 accounts.

✓

2

How are my sports teams doing?

Your last payday was on the 1st of November.

✗

3

Who has the best record in the NBA?

Sorry, I can only answer questions about banking.

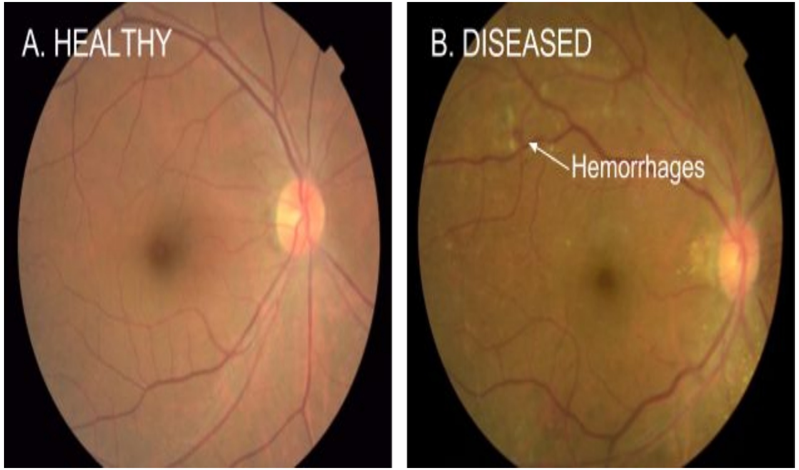
✓

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.

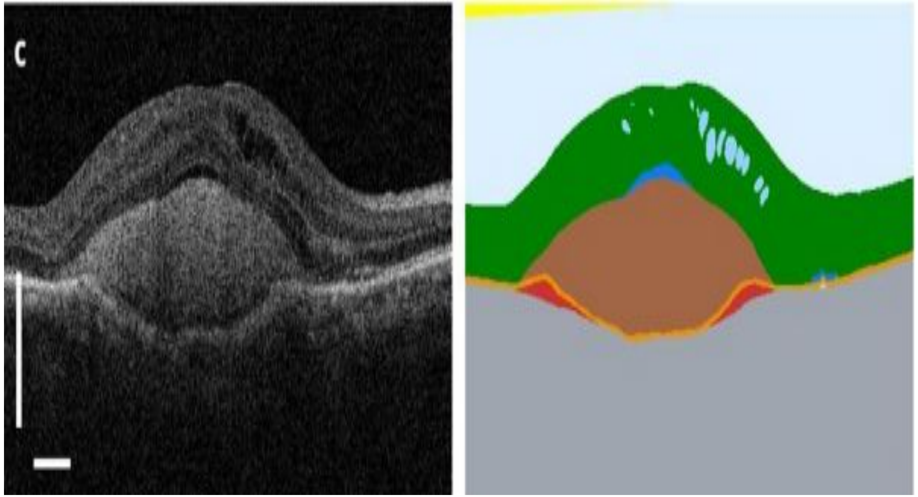
Image source: Larson et al. 2019 "An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction"

Medical Imaging

- Use model uncertainty to decide when to trust model vs when to defer to human.
- Reject out-of-distribution inputs.



Diabetic retinopathy detection from fundus images
Gulshan et al, 2016



Eye disease classification from 3D OCT images
de Fauw et al, 2018

Bayesian Optimization and Experimental Design

Google AI
Brain Team

- Exploration vs exploitation
- Use uncertainty for deciding tradeoff via acquisition

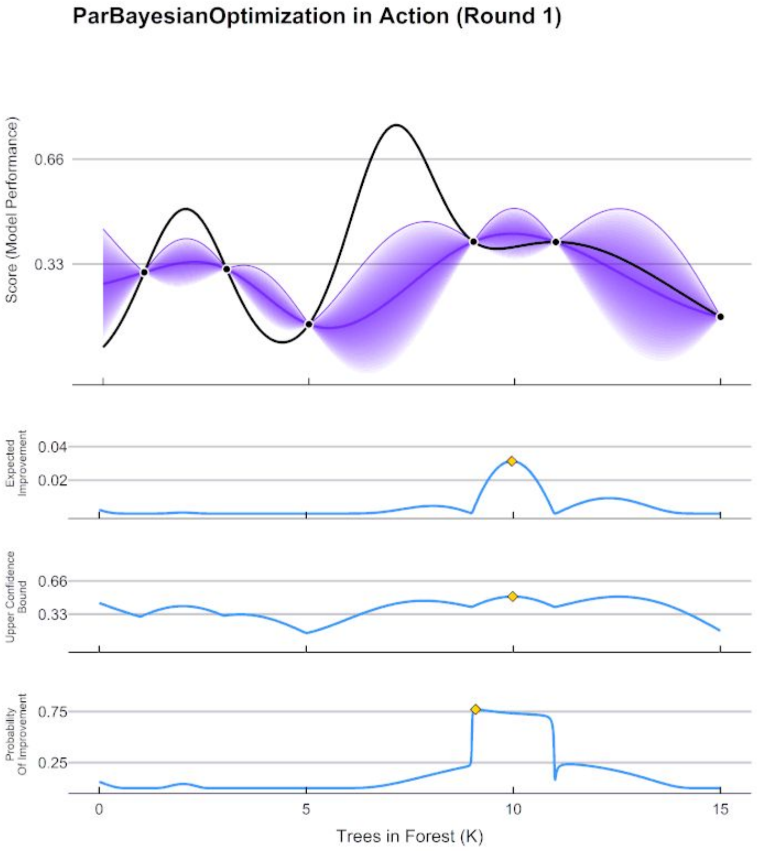
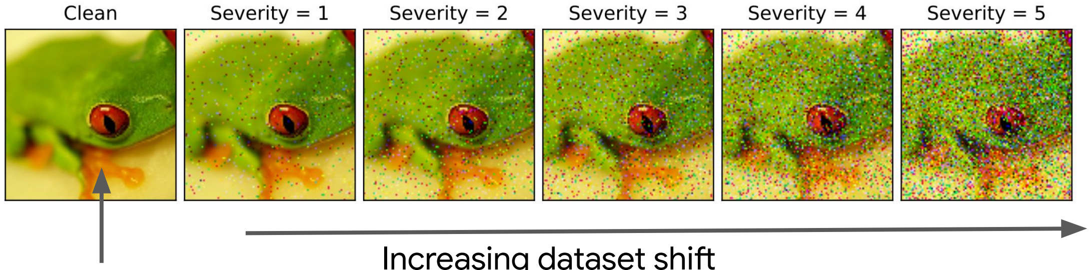


Image source:
https://en.wikipedia.org/wiki/Bayesian_optimization

How do current deep learning models fare?

ImageNet-C: Varying Intensity for Dataset Shift

- Typically we assume training and test data are i.i.d. from the same distribution



Increasing dataset shift

I.I.D test set

- In practice, often violated for test data and distributions shift

- ImageNet-C: different types of corruptions with varying intensity

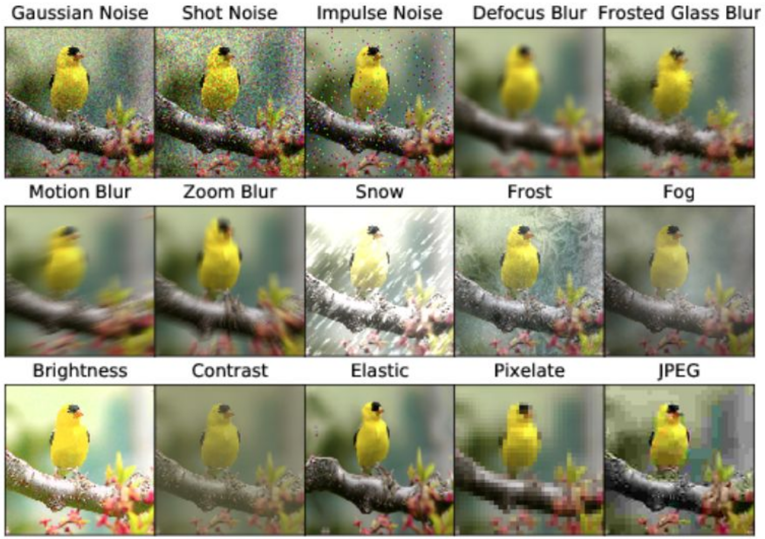
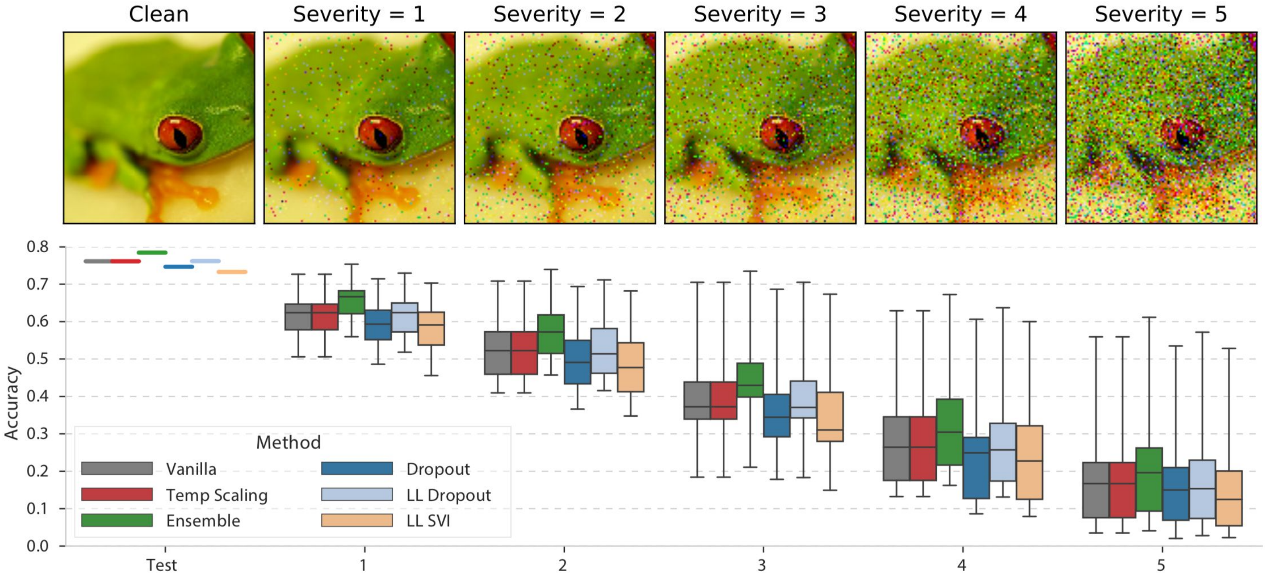


Image source: Benchmarking Neural Network Robustness to Common Corruptions and Perturbations, Hendrycks et al.

Models accuracy degrades under dataset shift

Google AI Brain Team

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



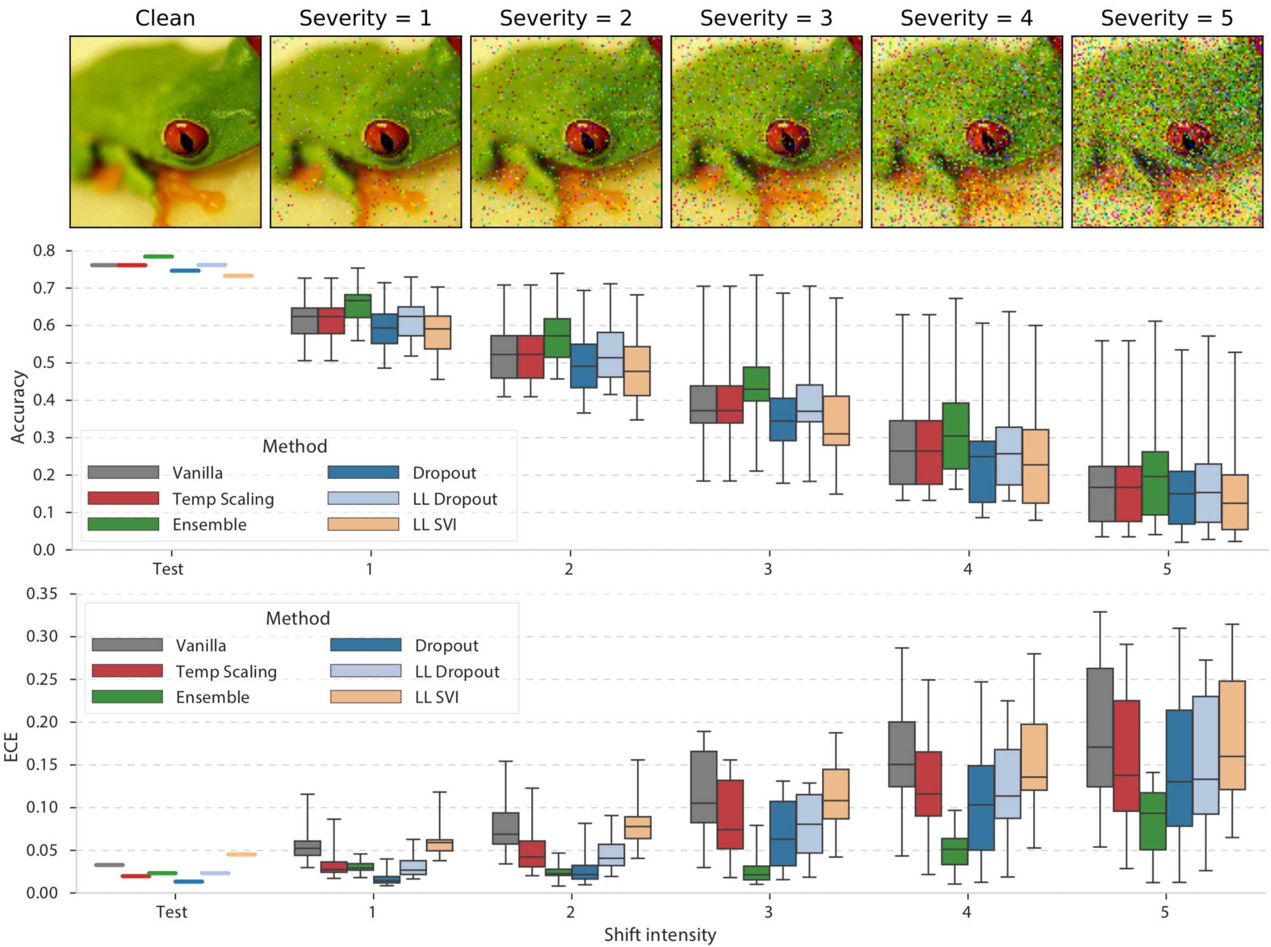
- But do the models know that they are less accurate?

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

Models are not calibrated under dataset shift

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

- **Calibration degrades with shift** -> “overconfident mistakes”



Models assign high confidence predictions to OOD inputs

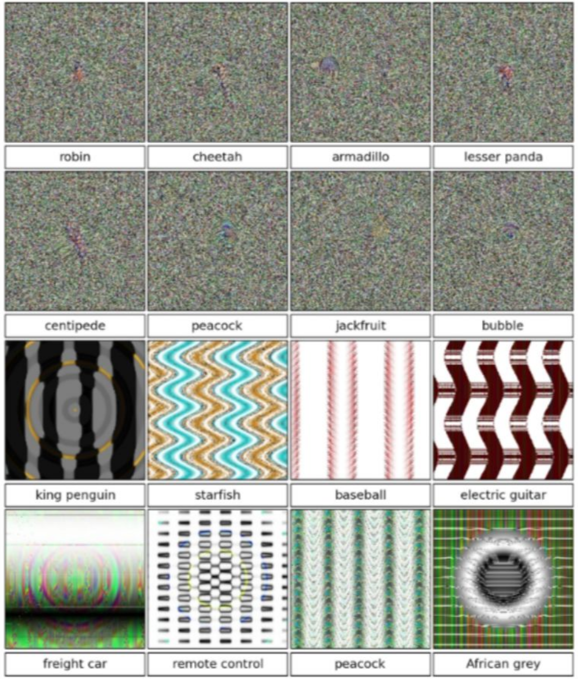
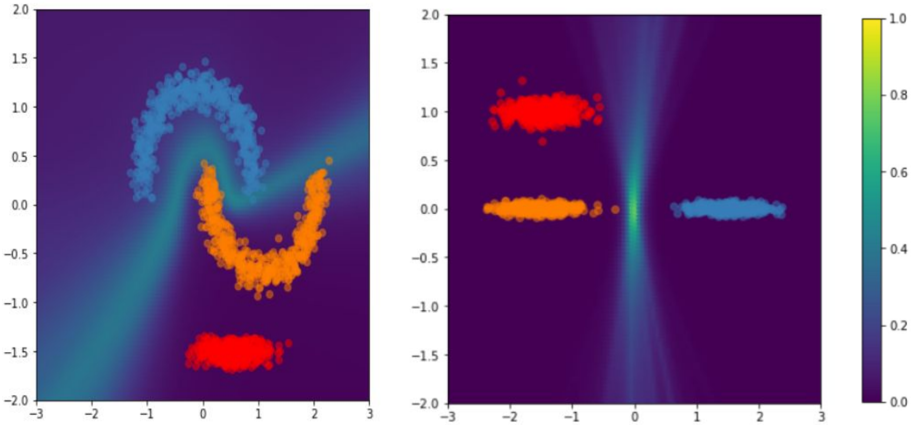


Figure 1. Evolved images that are unrecognizable to humans, but that state-of-the-art DNNs trained on ImageNet believe with $\geq$ 99.6% certainty to be a familiar object. This result highlights differences between how DNNs and humans recognize objects. Images are either directly (*top*) or indirectly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

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



High uncertainty (low confidence)

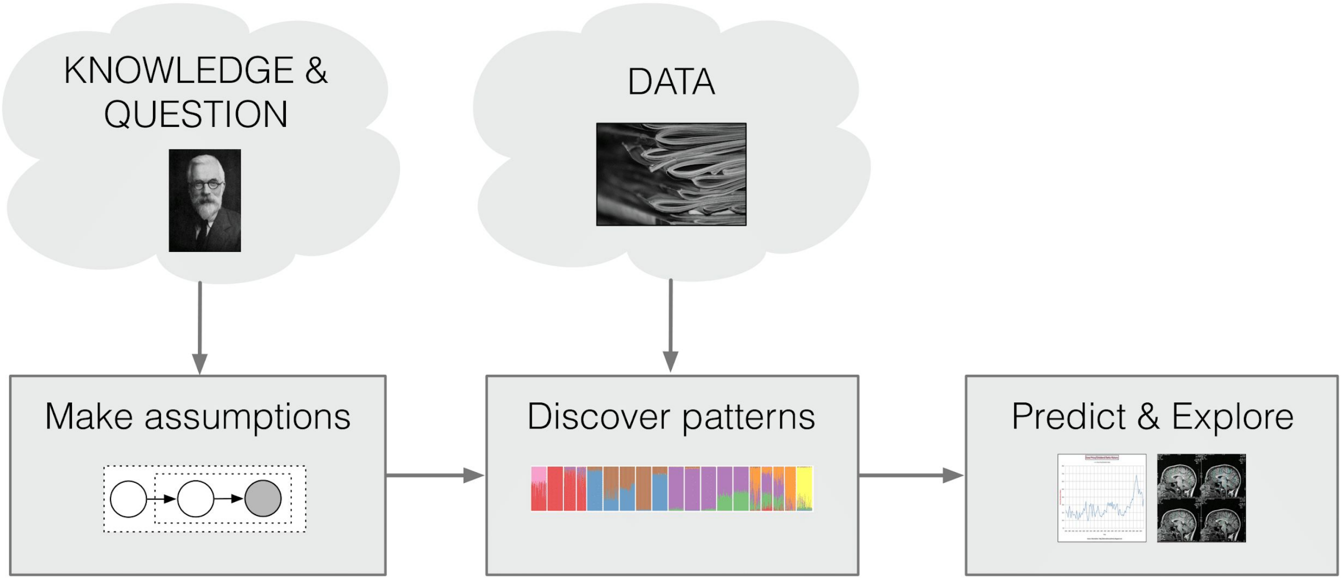
Low uncertainty (high confidence)

Illustration on toy binary classification (blue and orange) showing vanilla deep networks can assign high confidence to OOD inputs (red)

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

The Probabilistic Approach

The probabilistic approach



KNOWLEDGE & QUESTION

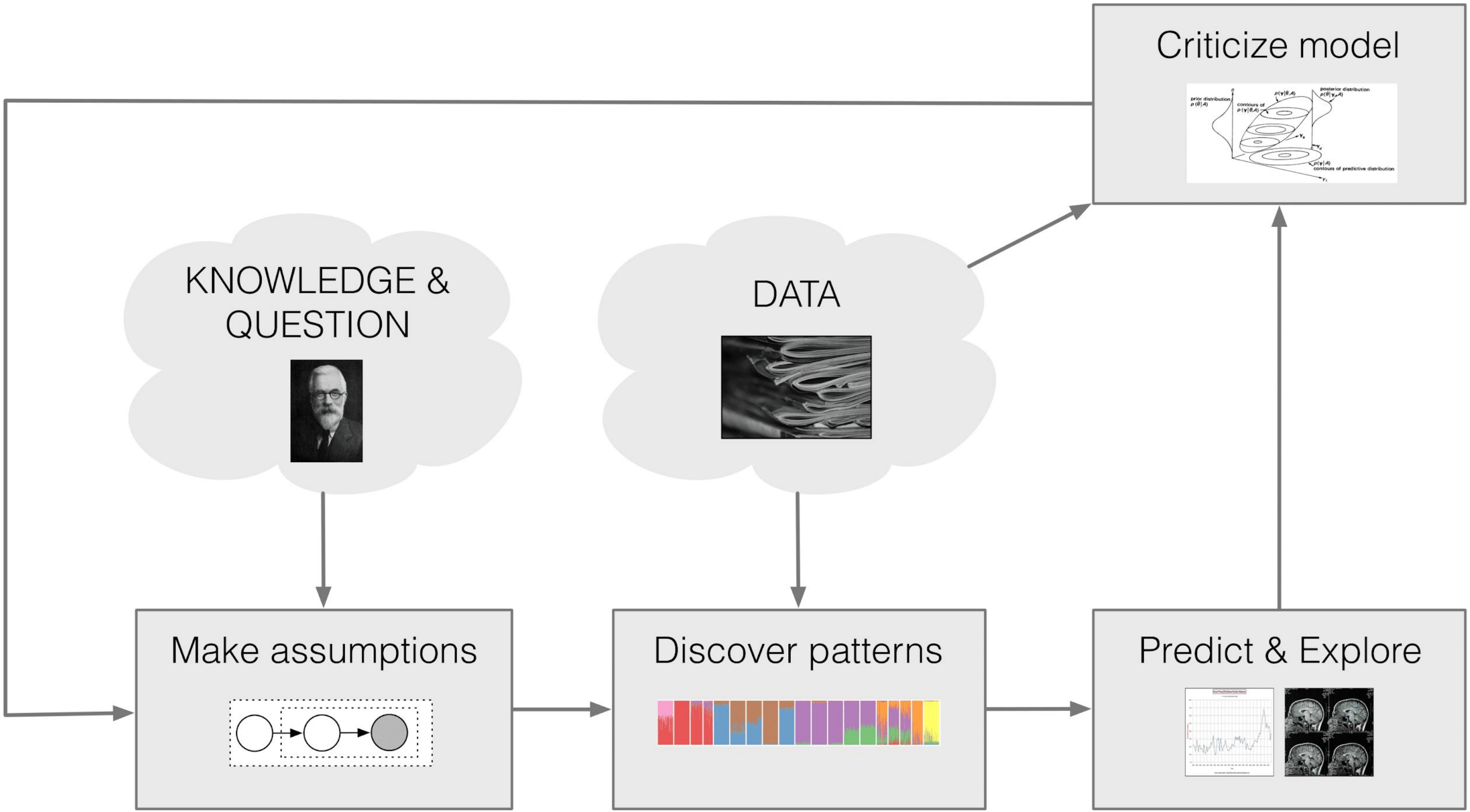
DATA

Make assumptions

Discover patterns

Predict & Explore

- Customized data analysis is important to many fields.
- Pipeline separates **assumptions**, **computation**, **application**
- Eases collaborative solutions to statistics problems



KNOWLEDGE & QUESTION

DATA

Make assumptions

Discover patterns

Predict & Explore

Criticize model

[Box, 1980; Rubin, 1984; Gelman+ 1996; Blei, 2014]

Probabilistic machine learning

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

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

Inference about the unknowns is through the **posterior**, the conditional distribution of the parameters given observations

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

For most interesting models, the denominator is not tractable. We appeal to **approximate posterior inference**.

Recipe for the probabilistic approach

1. Specify likelihood (neural net & output distribution) and prior.

2. Choose approximate inference procedure.
   - Variational approximation
   - MCMC
   - Ensembles

3. At test time, average predictions analytically or using samples from posterior.

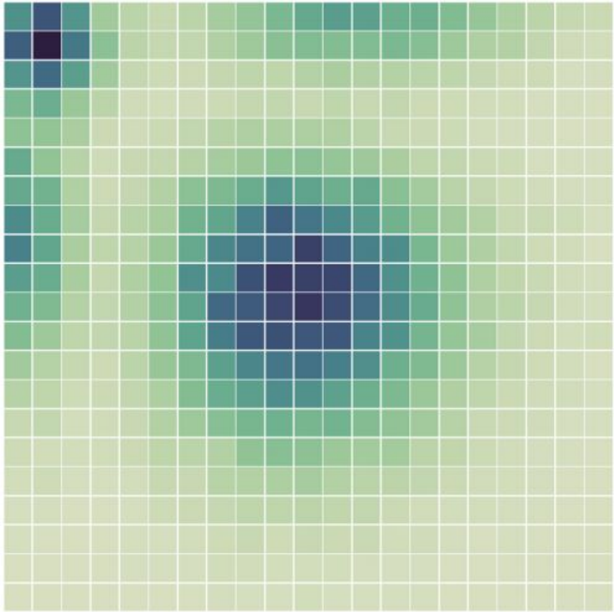
$$p(y|x, \mathcal{D}) \approx \frac{1}{S} \sum_{s=1}^{S} p(y|x, \boldsymbol{\theta}^{(s)})$$

Neural Networks with SGD

Google AI Brain Team

A simple approach is to use a point to approximate the posterior distribution. Select the parameters that attain highest probability under the distribution.

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



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

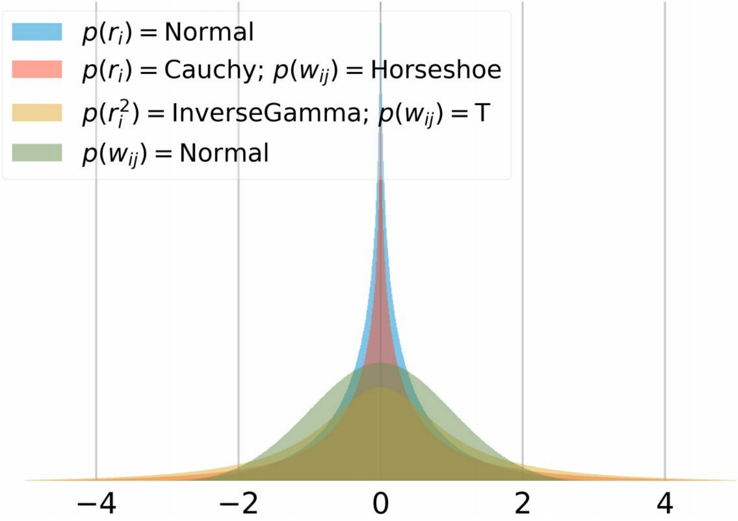
Methods

Bayesian Neural Networks

Google AI Brain Team

Two extra ingredients to neural nets w/ SGD:

1. Prior $p(\boldsymbol{\theta})$.



2. Family of distributions $q(\boldsymbol{\theta}; \boldsymbol{\lambda})$ to approximate the true posterior.

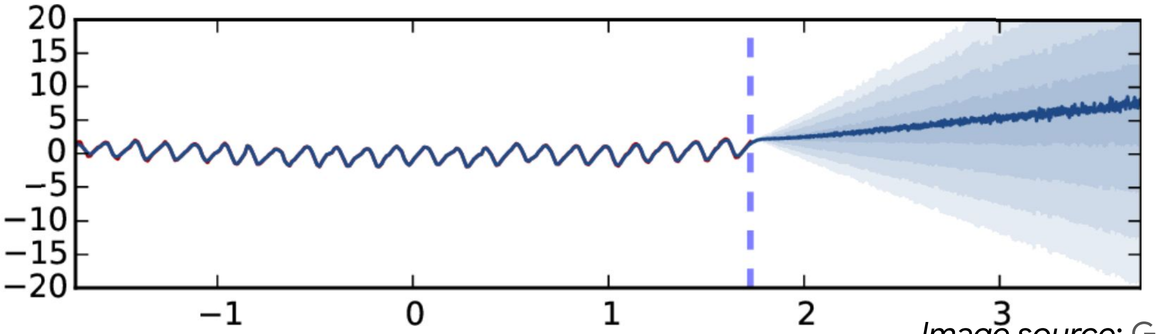
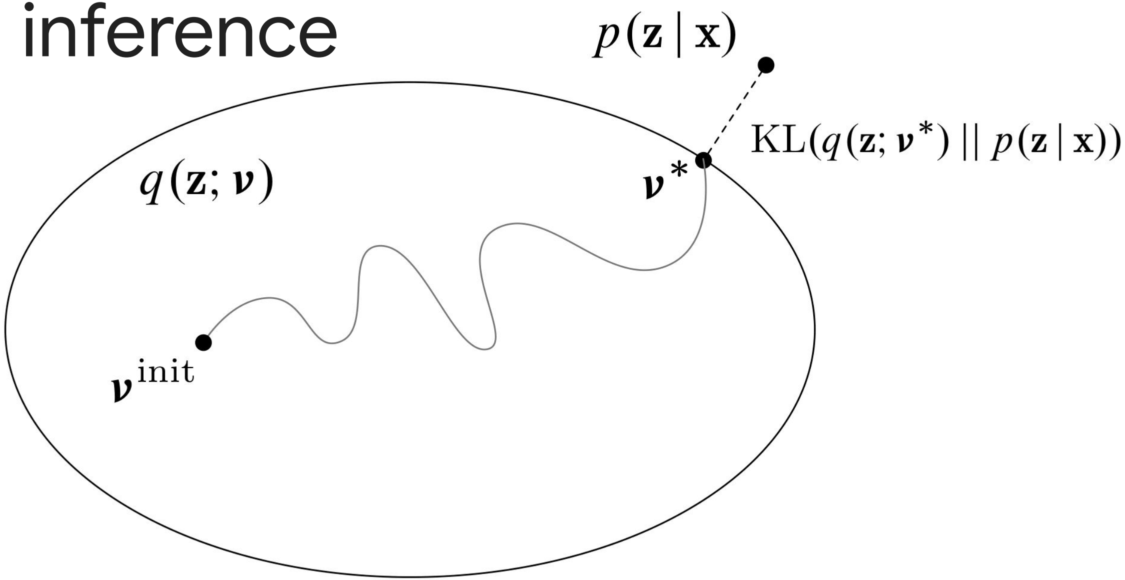


Image source: Gal+ 2015, Dusenberry+ 2020.

Variational inference


$p(\mathbf{z} \mid \mathbf{x})$
$\mathrm{KL}(q(\mathbf{z}; \boldsymbol{\nu}^*) \,||\, p(\mathbf{z} \mid \mathbf{x}))$
$q(\mathbf{z}; \boldsymbol{\nu})$
$\boldsymbol{\nu}^*$
$\boldsymbol{\nu}^{\text{init}}$

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

Image source: Blei Mohamed Ranganath. NeurIPS tutorial 2014.

Loss function

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 known as the evidence lower bound (ELBO).

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

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

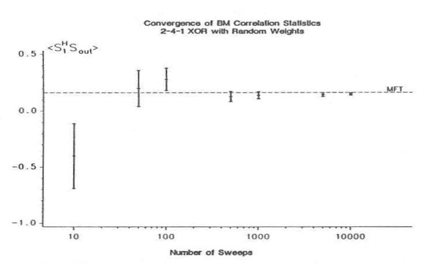
How do we select the prior?

Standard normal prior is the default. But.. it's not great.

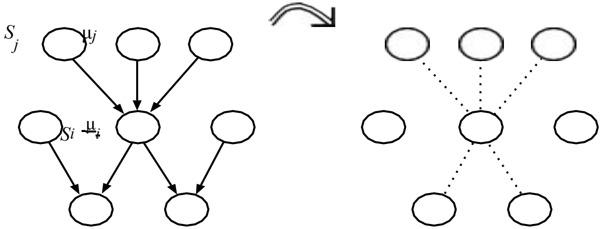
- It has bad statistical properties.
  - It does not leverage information about the network structure, unit-wise or layer-wise.
  - In the infinite-limit, all hidden units contribute infinitesimally to each input. [Neal 1994]
  - It's not clear how to improve the prior for specific properties, e.g., exploration.
- It has bad optimization properties.
  - It is sensitive to parameterization.
  - It's too strong a regularizer. The gradient signal for moving toward Normal(0, 1) dominates actually fitting the data. [eg Bowman+ 2015; Trippe Turner 2018]

Arguably, we have more intuition about priors in function space. [Hafner+ 2018, Sun+ 2019; Wang+ 2019; Louizos+ 2019]

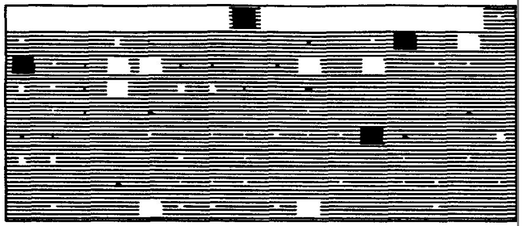
How do we select the approximate posterior?



[Peterson and Anderson 1987]



[Jordan et al. 1999]



[Hinton and van Camp 1993]

- VI began in 80's fitting probabilistic models with neural nets. [Peterson & Anderson 1987; Hinton & Van Camp 1993; Saul+ 1995].
  Used a mean-field distribution $q(\mathbf{z}; \boldsymbol{\lambda}) = \prod_{i=1}^{a} q(z_i; \boldsymbol{\lambda}_i)$.
- Mixture of mean-field distributions captures multimodality. [Jaakkola & Jordan 1998; Jordan+ 1999; Lawrence 2000]
- Structured factorizations maintain specific dependencies. [Saul & Jordan 1995; Barber & Wiegerinck 1999]

Image source: Blei, Mohamed, Ranganath. NeurIPS tutorial 2014.

Markov Chain Monte Carlo

We can approximate the posterior predictive via Monte Carlo

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

$$p(y|x,\mathcal{D}) \approx \tfrac{1}{S} \textstyle\sum_{s=1}^{S} p(y|x,\boldsymbol{\theta}^{(s)})$$

MCMC is a classic method to draw samples $\boldsymbol{\theta}^{(s)}$ from $p(\boldsymbol{\theta}|\mathcal{D})$ by only evaluating the *posterior energy*

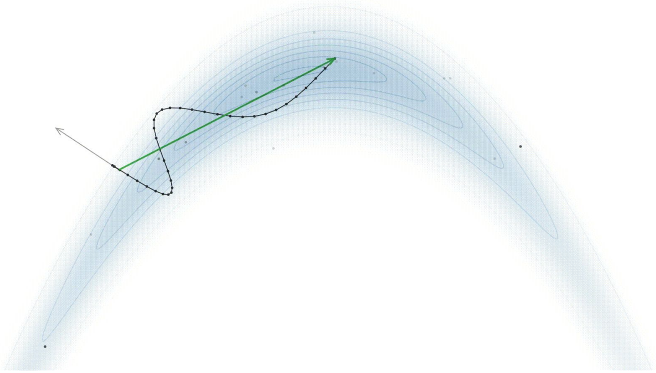
$$U(\boldsymbol{\theta}) := -\sum_{i=1}^{n} \log p(y_i|x_i,\boldsymbol{\theta}) - \log p(\boldsymbol{\theta})$$

via e.g. a *carefully* guided random walk in $\boldsymbol{\theta}$.

MCMC for Neural Networks

Adopted to neural nets from statistical physics - Neal, 94

- Hamiltonian Monte Carlo (Neal, 94)
    - Often cited as "gold standard" for Bayesian neural networks
    - Full-batch gradient descent with random initial momentum
- Langevin Dynamics
    - A single step of GD (Neal, 94)
    - Stochastic Gradient Langevin Dynamics (Welling & Teh, 2011)
- *Lots* of literature on different methods and scaling
    - Bayesian Inference for Large Scale Image Classification (Heek & Kalchbrenner, 2020)
    - Cyclical stochastic gradient MCMC for Bayesian deep learning (Zhang et al, 2020)
    - And many more...
- Caveats
    - Typically requires tricks to make it work - see Wenzel et al., 2020
    - Impractical - requires many samples
        - Can we carry around thousands of copies of a ResNet?



Hamiltonian Monte Carlo Demo

From https://github.com/chi-feng/mcmc-demo

Introduction to MCMC for Deep Learning, Iain Murray

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. Optimize T on a separate “recalibration” dataset.

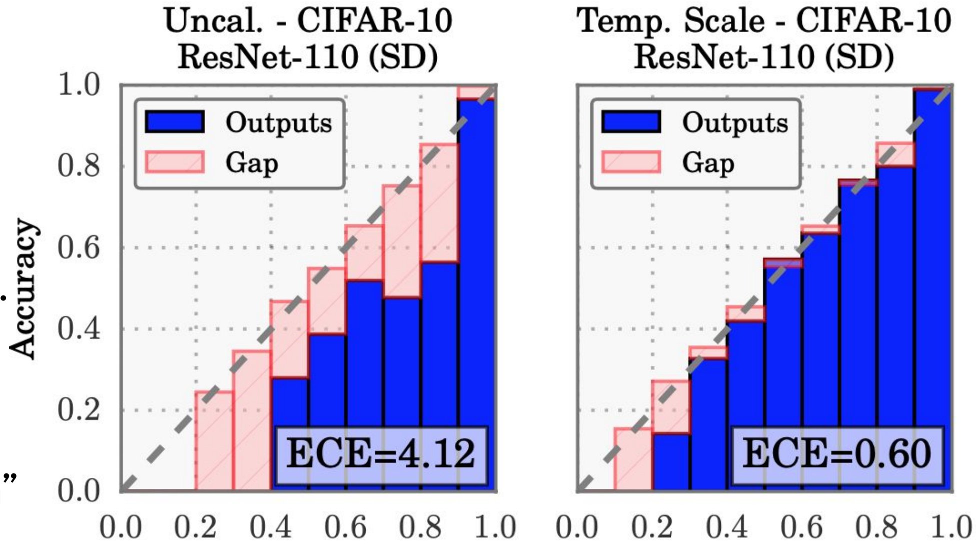


Image source: Guo+ 2017 “On calibration of modern neural networks”

Simple Baseline: Monte Carlo Dropout

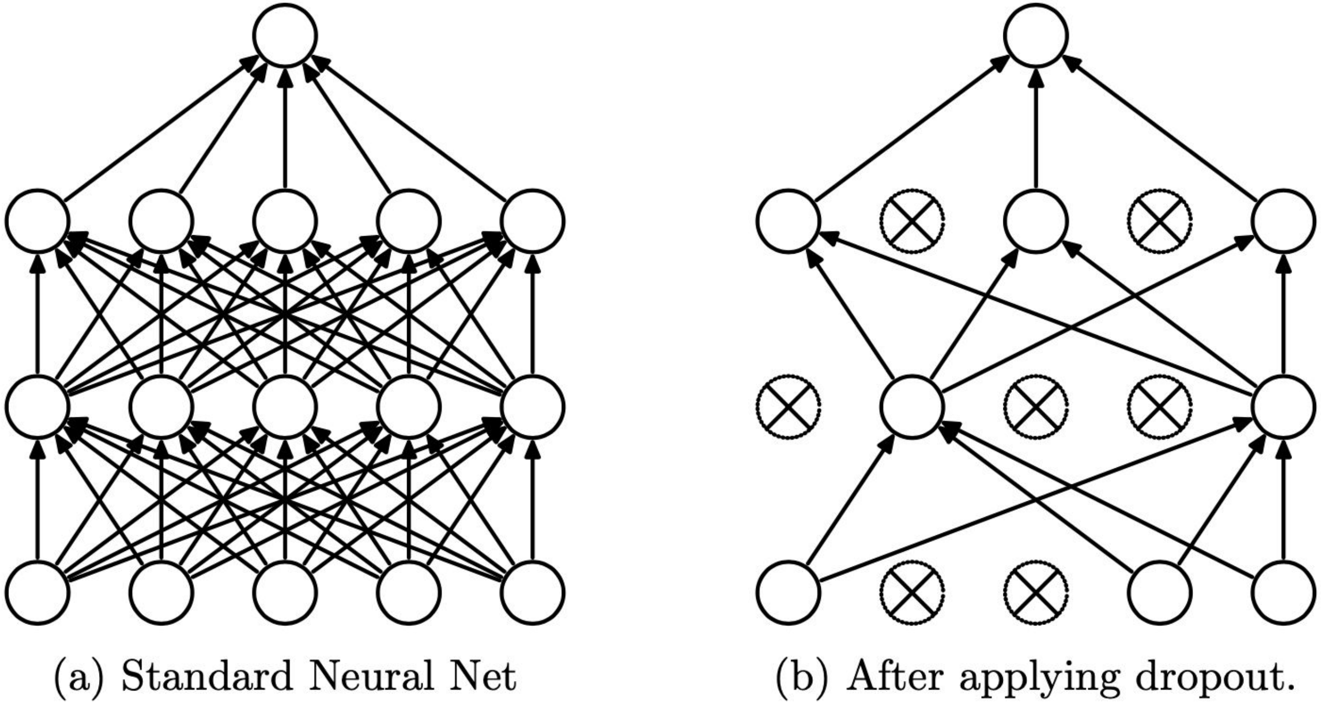

(a) Standard Neural Net
(b) After applying dropout.

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

[Gal+ 2015]

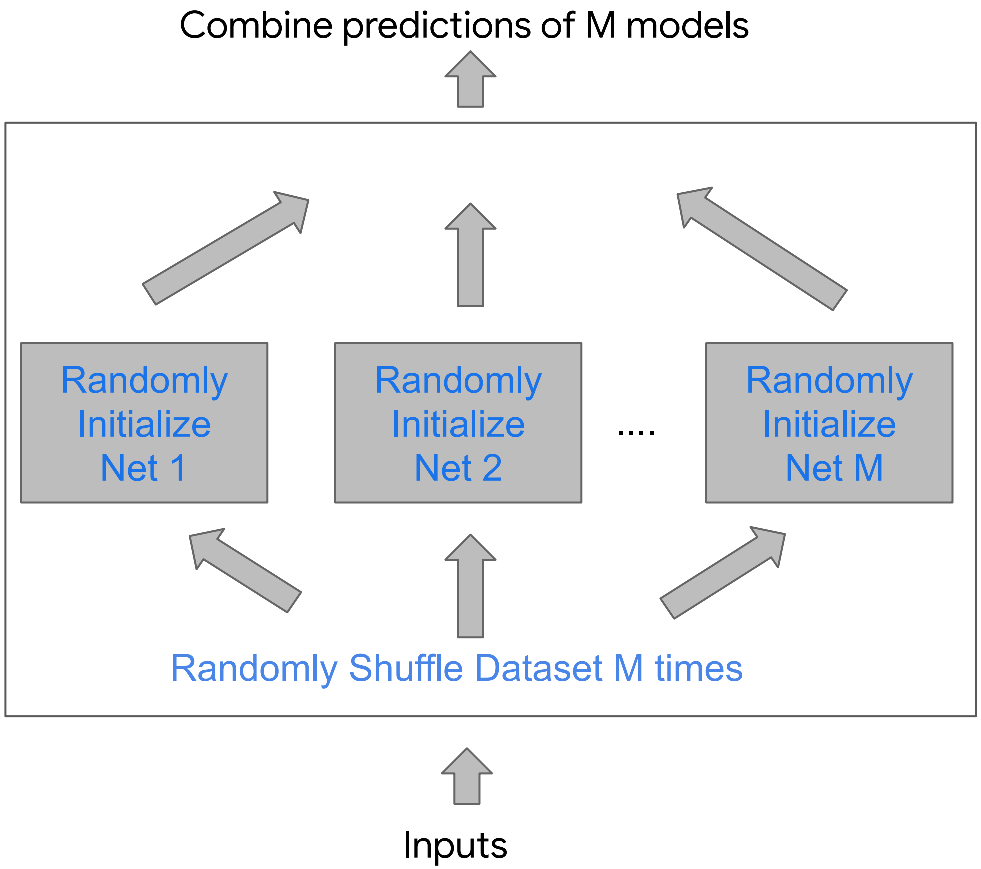
Simple Baseline: Deep Ensembles

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

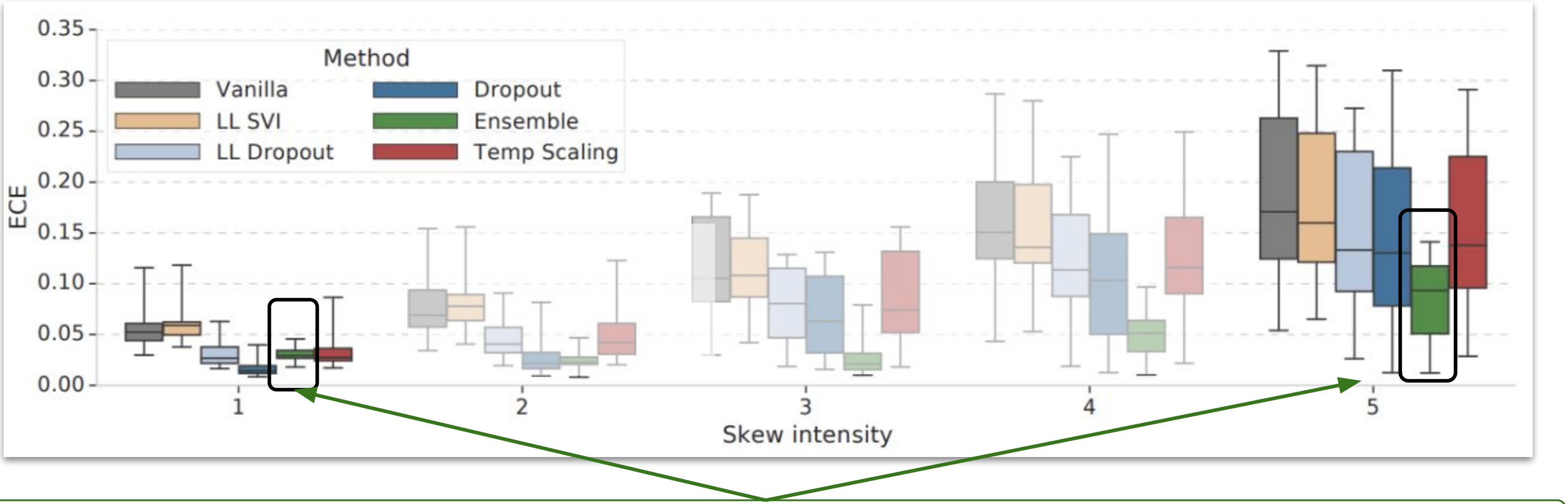
- A well known trick for getting better accuracy and Kaggle scores
- We rely on the fact that the loss landscape is non-convex to land at different solutions
  - Rely on different initializations and SGD noise

Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles, Lakshminarayanan et al.

- Found that the uncertainty produced by an ensemble is surprisingly good


Combine predictions of M models
Randomly Initialize Net 1
Randomly Initialize Net 2
....
Randomly Initialize Net M
Randomly Shuffle Dataset M times
Inputs

Deep Ensembles work surprisingly well in practice



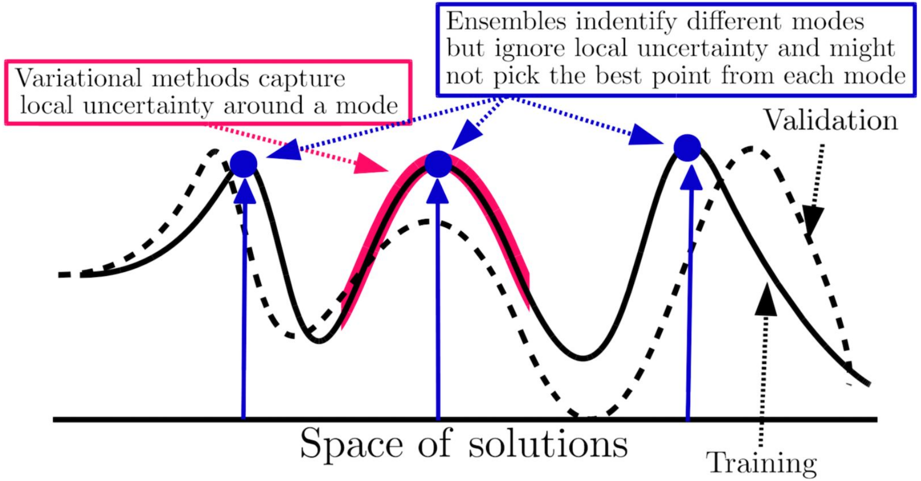
Deep Ensembles are consistently among the best performing methods, especially under dataset shift

Why do deep ensembles work well in practice?

Variational Bayesian methods are effective at **averaging uncertainty within a single mode**, but **fail to explore the diversity of multiple modes**

Random init + SGD noise explores different modes in function space

Deep Ensembles: A loss landscape perspective, Fort et al.


Ensembles indentify different modes but ignore local uncertainty and might not pick the best point from each mode
Variational methods capture local uncertainty around a mode
Validation
Space of solutions
Training

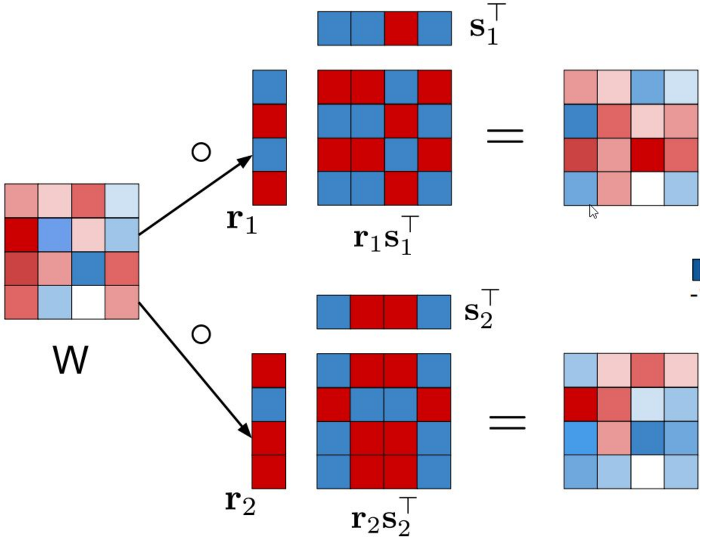
But… what about compute?

An ensemble's cost for both training and testing increases linearly with the number of networks. This becomes untenable for large models.

Bayesian neural nets show promise for improved uncertainty estimates (and capture different behavior). But they underfit at scale, and are also parameter inefficient!

How can we address these challenges?

BatchEnsemble



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.

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

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

The model yields *K* outputs for each example.

[Wen+ 2020]

Rank-1 Bayesian NNs

Combine efficient ensembles with Bayesian NNs!

Rank-1 BNNs:

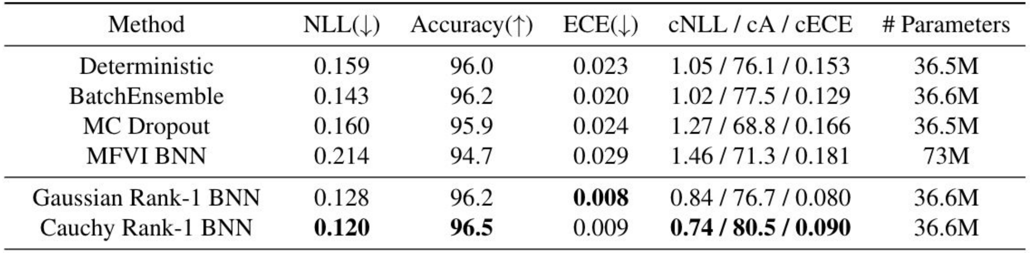
1. Start from BatchEnsemble's parameterization.
2. Add priors over rank-1 weights p(**r**), p(**s**).

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

3. Use global mixture variational posteriors.

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

Bayesian NNs struggle with underfitting at scale & parameter inefficiency. Rank-1 BNNs aims to solve both.



Method	NLL(↓)	Accuracy(↑)	ECE(↓)	cNLL / cA / cECE	# Parameters
Deterministic	0.159	96.0	0.023	1.05 / 76.1 / 0.153	36.5M
BatchEnsemble	0.143	96.2	0.020	1.02 / 77.5 / 0.129	36.6M
MC Dropout	0.160	95.9	0.024	1.27 / 68.8 / 0.166	36.5M
MFVI BNN	0.214	94.7	0.029	1.46 / 71.3 / 0.181	73M
Gaussian Rank-1 BNN	0.128	96.2	**0.008**	0.84 / 76.7 / 0.080	36.6M
Cauchy Rank-1 BNN	**0.120**	**96.5**	0.009	**0.74 / 80.5 / 0.090**	36.6M

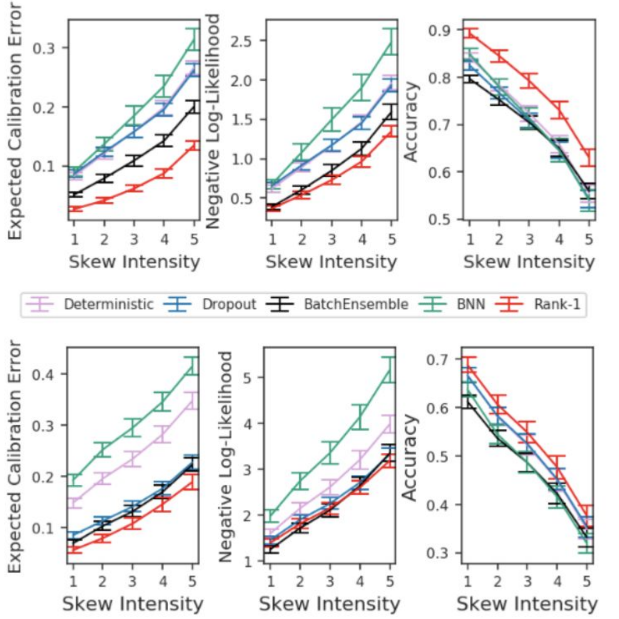


Figure 9: Out-of-distribution performance using CIFAR-10-C (**top**) and CIFAR-100-C (**bottom**) with WRN-28-10.

[Dusenberry+ 2020]

Google AI
Brain Team

Gaussian Processes

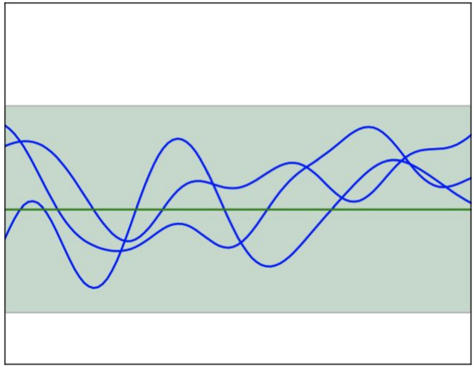
We can compute the integral $p(y|x, \mathcal{D}) = \int p(y|x, \boldsymbol{\theta})\, p(\boldsymbol{\theta}|\mathcal{D})\, \mathrm{d}\boldsymbol{\theta}$ analytically!

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

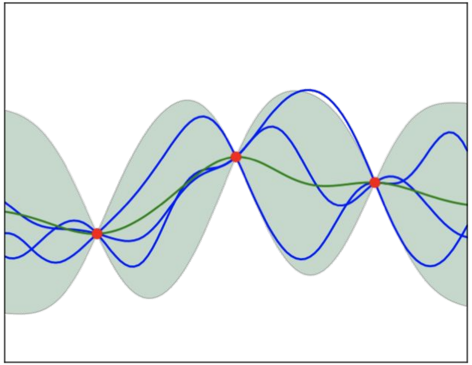
The result is a flexible distribution over functions

- Specified now by a covariance function over examples
  - Familiar with the kernel trick?
- Get a posterior on functions conditioned on data

See Rasmussen & Williams, 2006



Prior



Posterior

Gaussian Processes

Distribution over functions $f : \mathcal{X} \rightarrow \mathbb{R}$

The observations at points $\{\mathbf{x}_n \in \mathcal{X}\}_{n=1}^{N}$ are jointly Gaussian

Specified by a mean function $m : \mathcal{X} \rightarrow \mathbb{R}$ and covariance $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$

Predictive mean and covariance given observations:

$$\mu(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta) = K(\mathbf{X}, \mathbf{x})^\top K(\mathbf{X}, \mathbf{X})^{-1}(\mathbf{y} - m(\mathbf{X}))$$
$$\Sigma(\mathbf{x}, \mathbf{x}'\,;\,\{\mathbf{x}_n, y_n\}, \theta) = K(\mathbf{x}, \mathbf{x}') - K(\mathbf{X}, \mathbf{x})^\top K(\mathbf{X}, \mathbf{X})^{-1} K(\mathbf{X}, \mathbf{x}')$$

Intuition:

- A prior for smooth functions
- Similar inputs (high covariance) should have a similar outputs
- Can compute expected value and uncertainty for a test input easily

Infinite Width Deep Neural Networks are Gaussian Processes

- In the limit of infinite width + Gaussian prior converges to a GP (Neal, 94)
  - i.e. covariance is taken over the hidden layer activations
- Computing with infinite networks Williams, 97
  - Derived a covariance function for single layer with “erf” activations
- Recently renewed interest
  - Deep Neural Networks as Gaussian Processes, Lee 2018
  - Gaussian process behaviour in wide deep neural networks, Matthews 2018
  - + many more.
- 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

Recent Work

AugMix improves calibration under shift

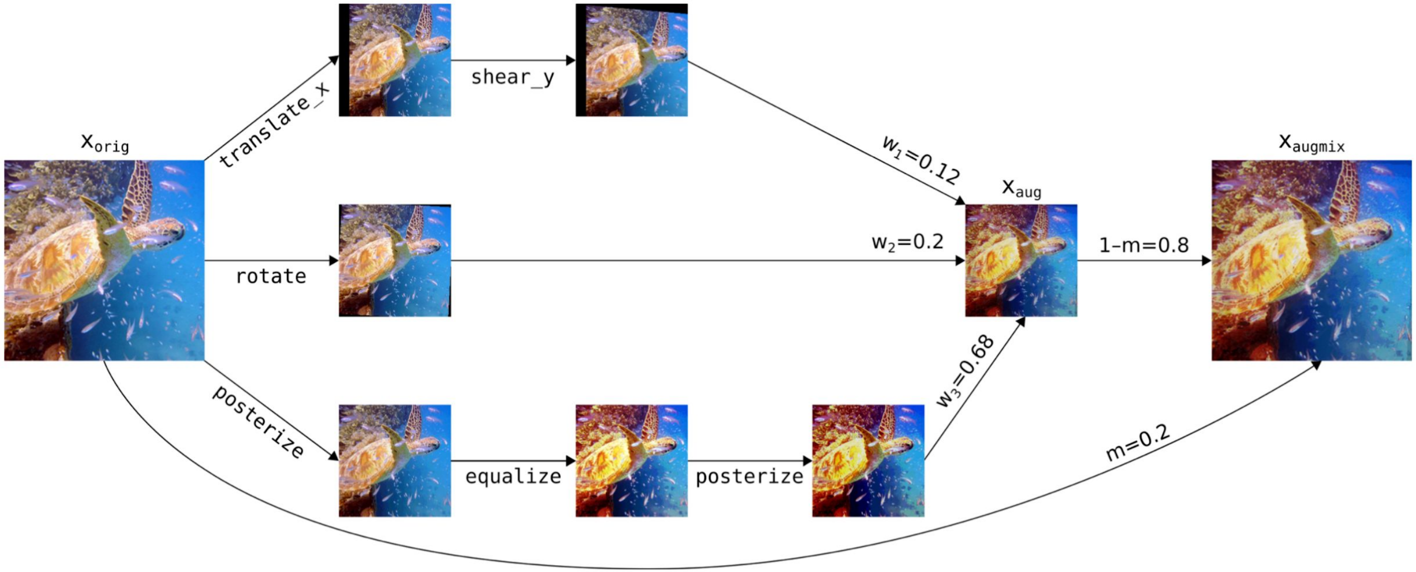
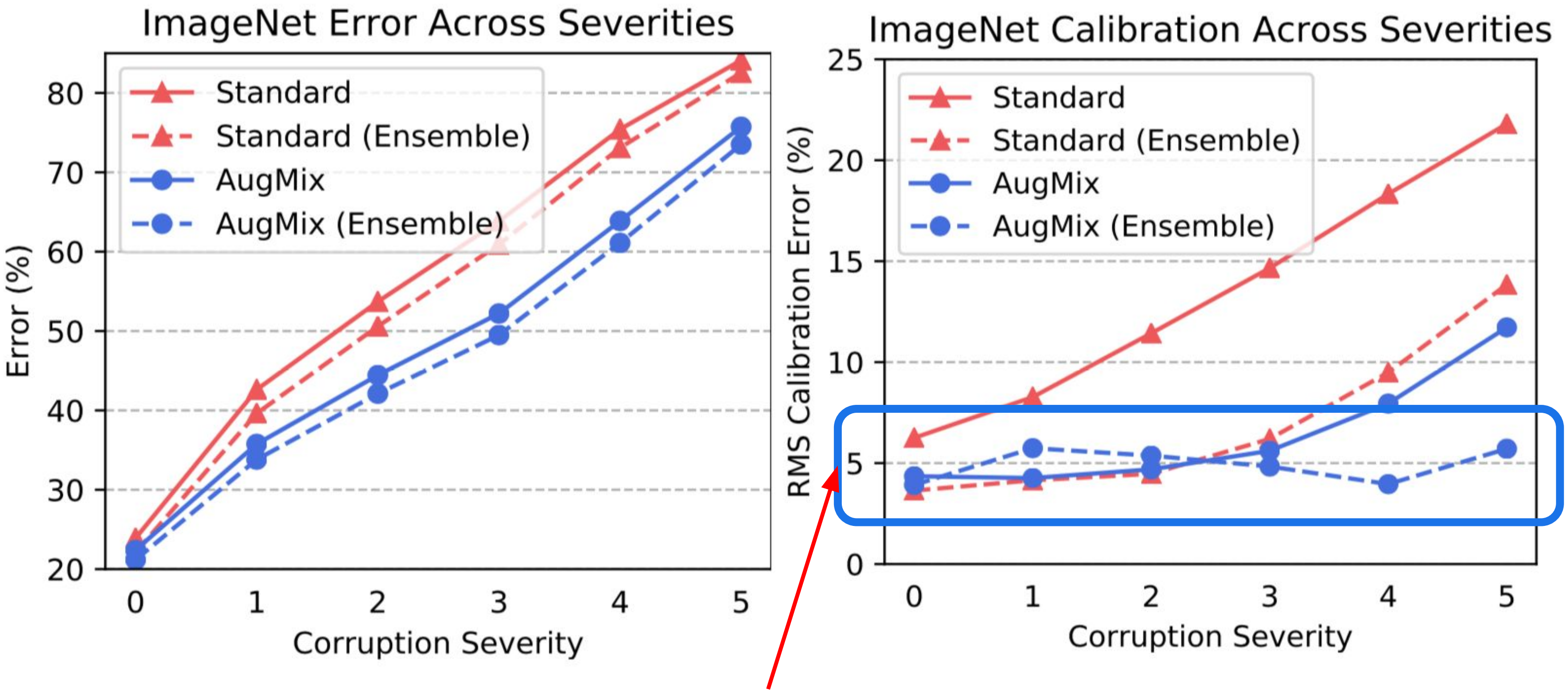


Figure 4: A realization of AUGMIX. Randomly sampled operations and their compositions allow us to explore the semantically meaningful input space around an image. Mixing these images together produces a new image without veering too far from the original.

Better data augmentation (composing base operations and 'mixing' them) and enforcing consistency can encode invariances and improve calibration under dataset shift. ([Hendrycks et al 2020](#))

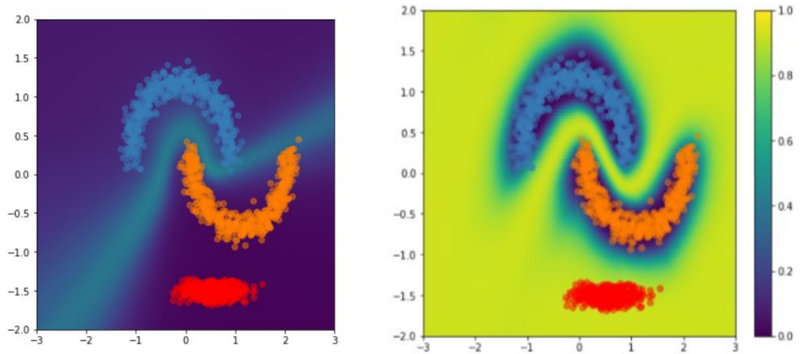
AugMix improves calibration under shift



AugMix + Deep Ensembles significantly improves calibration results under data shift ([Hendrycks et al 2020](#))
Data augmentation and self-supervised learning can provide complementary benefits to marginalization

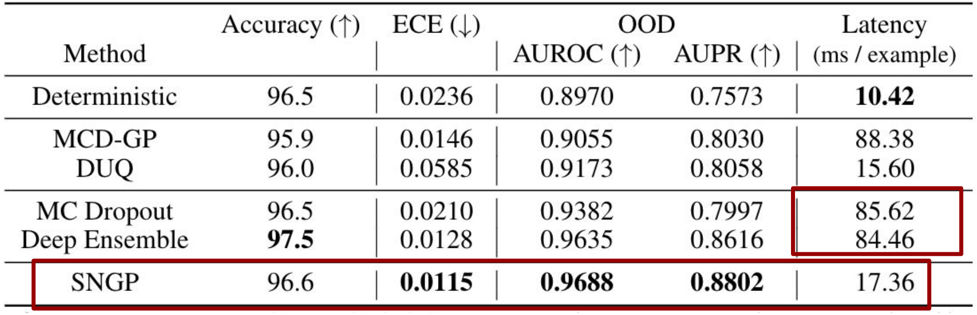
Improving “single model” uncertainty

- Spectral-normalized Neural Gaussian process (SNGP)
  - Replace last-layer with “GP layer”
  - SNGP uncertainty increases farther from training data.
  - “Distance-awareness” via biLipschitz constraint (spectral normalization)
  - SNGP outperforms softmax on OOD detection benchmarks (image & text)
- See also Deterministic Uncertainty Quantification (DUQ)



Deep Ensemble

SNGP (single model)

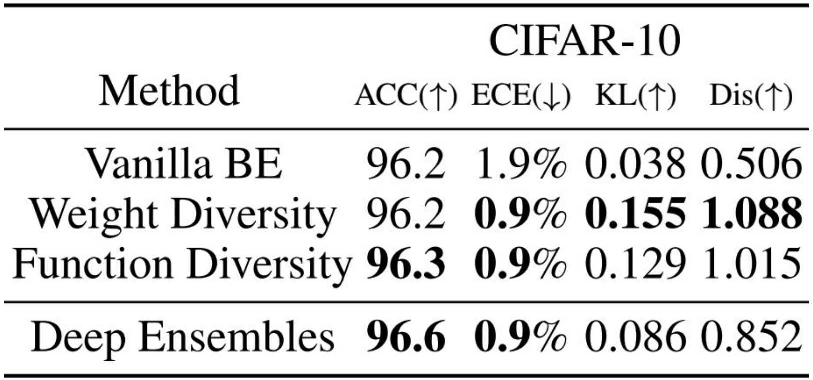


Method	Accuracy (↑)	ECE (↓)	OOD		Latency
			AUROC (↑)	AUPR (↑)	(ms / example)
Deterministic	96.5	0.0236	0.8970	0.7573	**10.42**
MCD-GP	95.9	0.0146	0.9055	0.8030	88.38
DUQ	96.0	0.0585	0.9173	0.8058	15.60
MC Dropout	96.5	0.0210	0.9382	0.7997	85.62
Deep Ensemble	**97.5**	0.0128	0.9635	0.8616	84.46
SNGP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

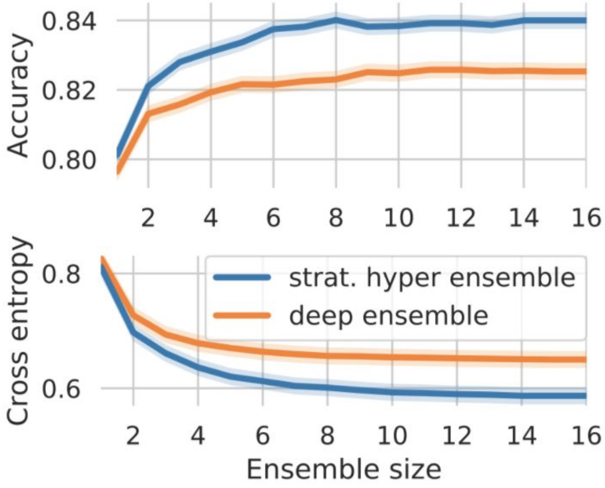
Results with BERT on intent detection benchmark

Diverse Ensembles

Vanilla deep ensembles differ only in random seeds.
Diverse ensembles achieve better trade-off on the uncertainty-compute frontier.



Method	CIFAR-10			
	ACC(↑)	ECE(↓)	KL(↑)	Dis(↑)
Vanilla BE	96.2	1.9%	0.038	0.506
Weight Diversity	96.2	**0.9%**	**0.155**	**1.088**
Function Diversity	**96.3**	**0.9%**	0.129	1.015
Deep Ensembles	**96.6**	**0.9%**	0.086	0.852

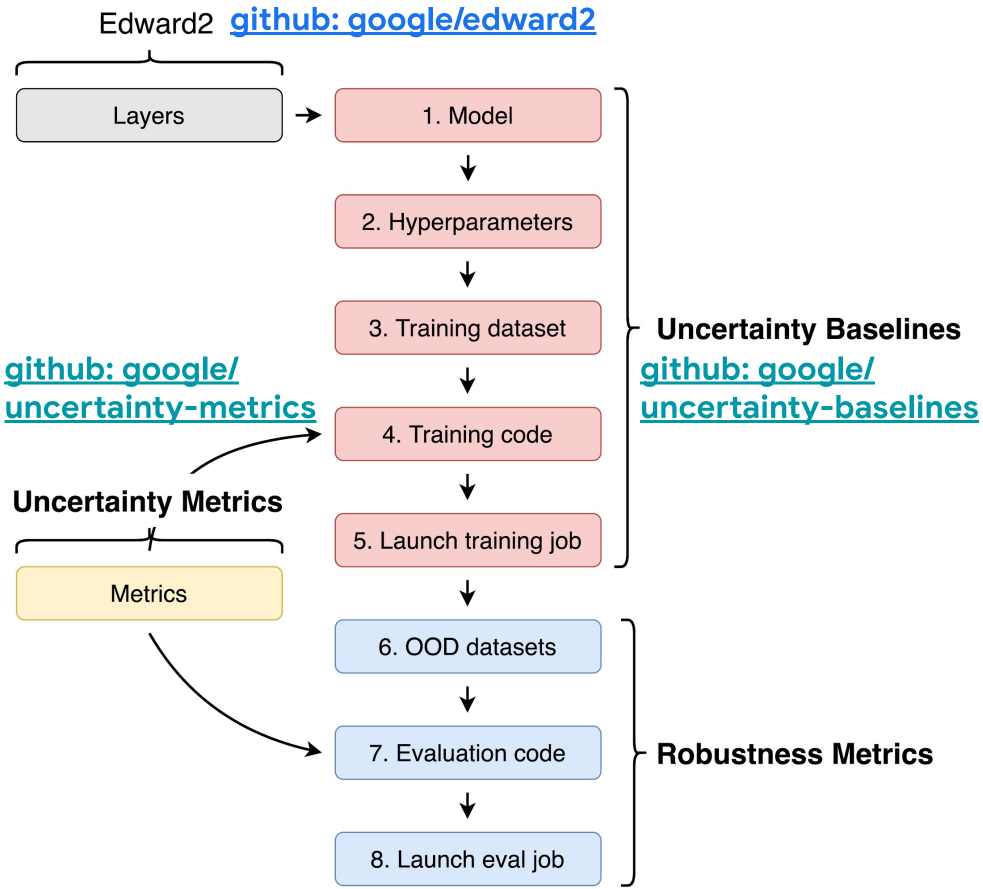


Adding diversity regularizer to Rank-1 ensembles

Hyperparameter ensembles outperform vanilla ensembles

Open Challenges

Code



Edward2 github: google/edward2

Layers

1. Model

2. Hyperparameters

3. Training dataset

4. Training code

5. Launch training job

Uncertainty Baselines

github: google/uncertainty-baselines

github: google/uncertainty-metrics

Uncertainty Metrics

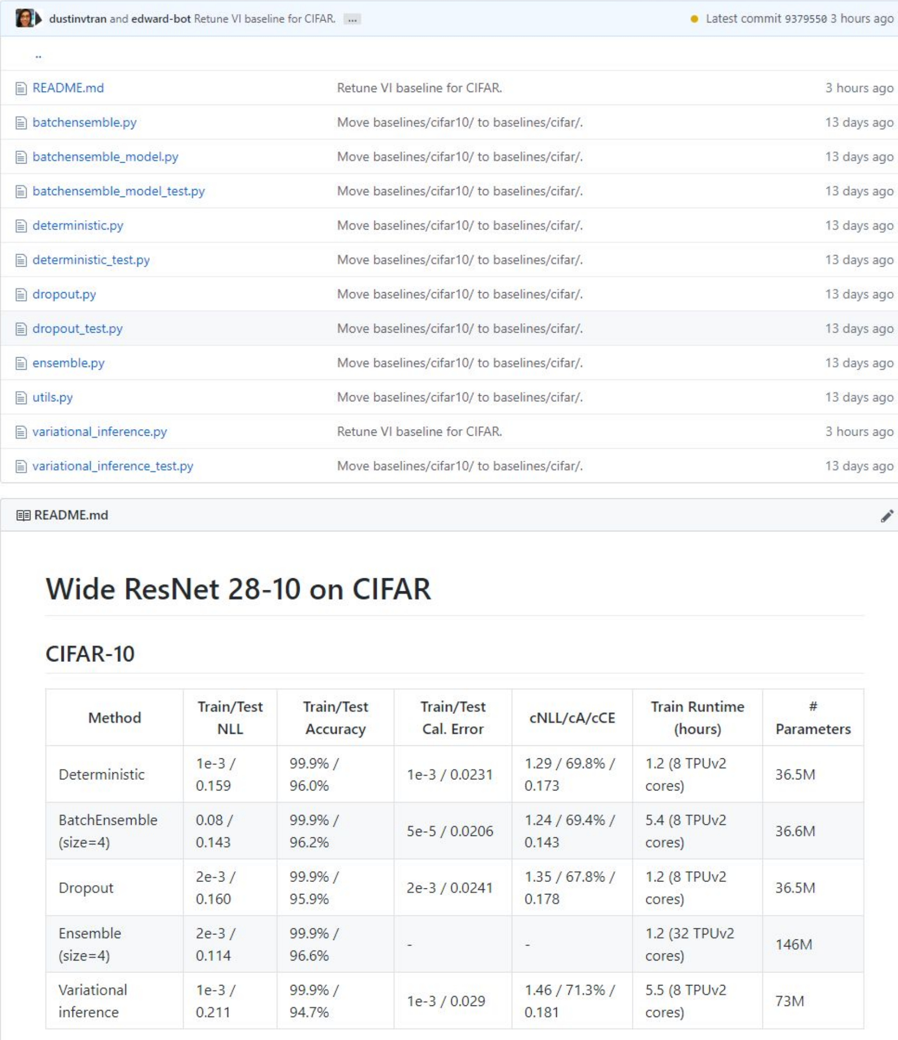
Metrics

6. OOD datasets

7. Evaluation code

8. Launch eval job

Robustness Metrics



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

Open Challenges

Probabilistic deep learning: Closing the gap between theory and practice

- How good is the Bayes posterior really, Wenzel et al. 2020?
- What are good priors over neural networks?
- What role does the choice of architecture, hyperparameters, and heuristics play?
- How do we efficiently marginalize over high-dimensional NN posteriors?
- Better understanding of out-of-distribution behavior of deep predictive models as well as deep generative models
- Is there a rigorous (Bayesian) interpretation of deep ensembles?

Realistic benchmarks that reflect real-world challenges

- Natural Distribution Shift on Question Answering Models [Miller+ 2020]
- Measuring Robustness to Natural Distribution Shifts in Image Classification [Taori 2020]

References

Probabilistic machine learning

- **Science and Statistics**. G. Box. JASA 1976.
- **Build, Compute, Critique, Repeat: Data Analysis with Latent Variable Models**. D. Blei. Annual Reviews 2014.

Bayesian neural networks

- **A practical Bayesian framework for backpropagation networks** D. MacKay Neural Computation 1992
- **Keeping Neural Networks Simple by Minimizing the Description Length of the Weights**. G. Hinton, D. Van Camp. COLT 1993.
- **An Introduction to Variational Methods for Graphical Models**. M. Jordan+. Machine Learning 1999.
- **Bayesian Learning for Neural Networks.** R. Neal. Technical Report 1994.
- **Bayesian Learning via Stochastic Gradient Langevin Dynamics**. M. Welling, Y. Teh. ICML 2011.
- **Weight Uncertainty in Neural Networks**. C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra. ICML 2015.
- **Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning** Y. Gal, Z. Ghahramani ICML 2016
- **Automatic Differentiation Variational Inference**. A. Kucukelbir, *D. Tran*, R. Ranganath, A. Gelman, D. M. Blei. JMLR 2017.
- **A Scalable Laplace Approximation for Neural Networks** H. Ritter, A. Botev, D. Barber ICLR 2018
- **Noise Contrastive Priors for Functional Uncertainty**. D. Hafner, *D. Tran*, T. Lillicrap, A. Irpan, J. Davidson. UAI 2019.
- **A Simple Baseline for Bayesian Uncertainty in Deep Learning** W. Maddox, T. Garipov, P. Izmailov, D. Vetrov, A. G. Wilson. NeurIPS 2019.
- **Practical Deep Learning with Bayesian Principles** K. Osawa, S. Swaroop, A.Jain, R. Eschenhagen, R. E. Turner, R. Yokota, M. E. Khan. NeurIPS 2019.
- **Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors**. M. W. Dusenberry, G. Jerfel, Y. Wen, Y. Ma, J. Snoek, K. Heller, B. Lakshminarayanan, *D. Tran*. ICML 2020.

References

Ensembles

- **Simple and scalable predictive uncertainty estimation using deep ensembles** *B. Lakshminarayanan*, A. Pritzel, C. Blundell. NeurIPS 2017.
- **BatchEnsemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning**. Y. Wen, *D. Tran*, J. Ba. ICLR 2020.

Understanding marginalization

- **Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data**. G. K. Dziugaite, D. M. Roy. UAI 2017.
- **Deep ensembles: A loss landscape perspective**. S. Fort, H. Hu, *B. Lakshminarayanan*. *arXiv 1912.02757*.
- **Bayesian Deep Learning and a Probabilistic Perspective of Generalization** A. G. Wilson and P. Izmailov arXiv 2002.08791

Gaussian processes and Neural Tangent Kernel

- **Deep Neural Networks as Gaussian Processes.** J. Lee, Y. Bahri, R. Novak, S. Schoenholz, J. Pennington, J. Sohl-Dickstein, ICLR 2018.
- **Neural Tangent Kernel: Convergence and Generalization in Neural Networks.** A. Jacot, F. Gabriel, C. Hongler. NeurIPS 2018.
- **Approximate Inference Turns Deep Networks into Gaussian Processes** M. Emtiyaz Khan, Alexander Immer, Ehsan Abedi, M. Korzepa NeurIPS 2019
- **Bayesian deep ensembles via the Neural Tangent Kernel** B. He, B. Lakshminarayanan, Y. W. Teh arXiv 2007.05864
- Exploring the Uncertainty Properties of Neural Networks' Implicit Priors in the Infinite-Width Limit. B. Adlam, J. Lee, L. Xiao, J. Pennington and J. Snoek link

References

Practical guidance

- See the codebases!
- **Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift**. Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D Sculley, S. Nowozin, J. Dillon, *B. Lakshminarayanan, J. Snoek*. NeurIPS 2019.
- **Simple, Distributed, & Accelerated Probabilistic Programming**. *D. Tran*, M. Hoffman, D. Moore, C. Suter, S. Vasudevan, A. Radul, M. Johnson, R. A. Saurous. NeurIPS 2018.
- **Bayesian Layers: A Module for Neural Network Uncertainty**. *D. Tran*, M. W. Dusenberry, M. van der Wilk, D. Hafner. NeurIPS 2019.

Data Augmentation and Invariances

- **Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty** *D. Hendrycks, M. Mazeika, S. Kadavath, D. Song NeurIPS 2019*
- **AugMix: A simple data processing method to improve robustness and uncertainty**. D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, *B. Lakshminarayanan. ICLR 2020*
- **Improving Calibration of BatchEnsemble with Data Augmentation**. *Y. Wen, G. Jerfel, R. Muller, M. Dusenberry, J. Snoek, B. Lakshminarayanan and D. Tran. link*
-

Calibration

- **On calibration of modern neural networks** C. Guo, G. Pleiss, Y. Sun, K. Q. Weinberger *ICML 2017*
- **Measuring Calibration in Deep Learning**. J. Nixon, M. Dusenberry, L. Zhang, G. Jerfel, *D. Tran*. arXiv 1904.01685

References

Proper Scoring Rules, Types of Uncertainty

- **Strictly Proper Scoring Rules, Prediction and Estimation**, *Gneiting & Raftery, JASA 2007*
- **What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?** *A. Kendall, Y. Gal NeurIPS 2017*
- **Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods** *E. Hüllermeier, W. Waegeman* arXiv 1910.09457

Deep Generative Models and Hybrid models

- **Hybrid models with deep and invertible features** E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, *B. Lakshminarayanan*. *ICML 2019*.
- **Do deep generative models know what they don't know?** E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, *B. Lakshminarayanan*. *ICLR 2019*.
- **Likelihood ratios for out-of-distribution detection.** J. Ren, P. Liu, E. Fertig, J. Snoek, R. Poplin, M. DePristo, J. Dillon, *B. Lakshminarayanan*. *NeurIPS 2019*.
- **Detecting out-of-distribution inputs to deep generative models using a test for typicality.** E. Nalisnick, A. Matsukawa, Y. W. Teh, *B. Lakshminarayanan*. *arXiv 1906.02994*.
- **Density of States Estimation for Out-of-Distribution Detection** W. R. Morningstar, C. Ham, A. G. Gallagher, B. Lakshminarayanan, A. A. Alemi, J. V. Dillon arXiv 2006.09273

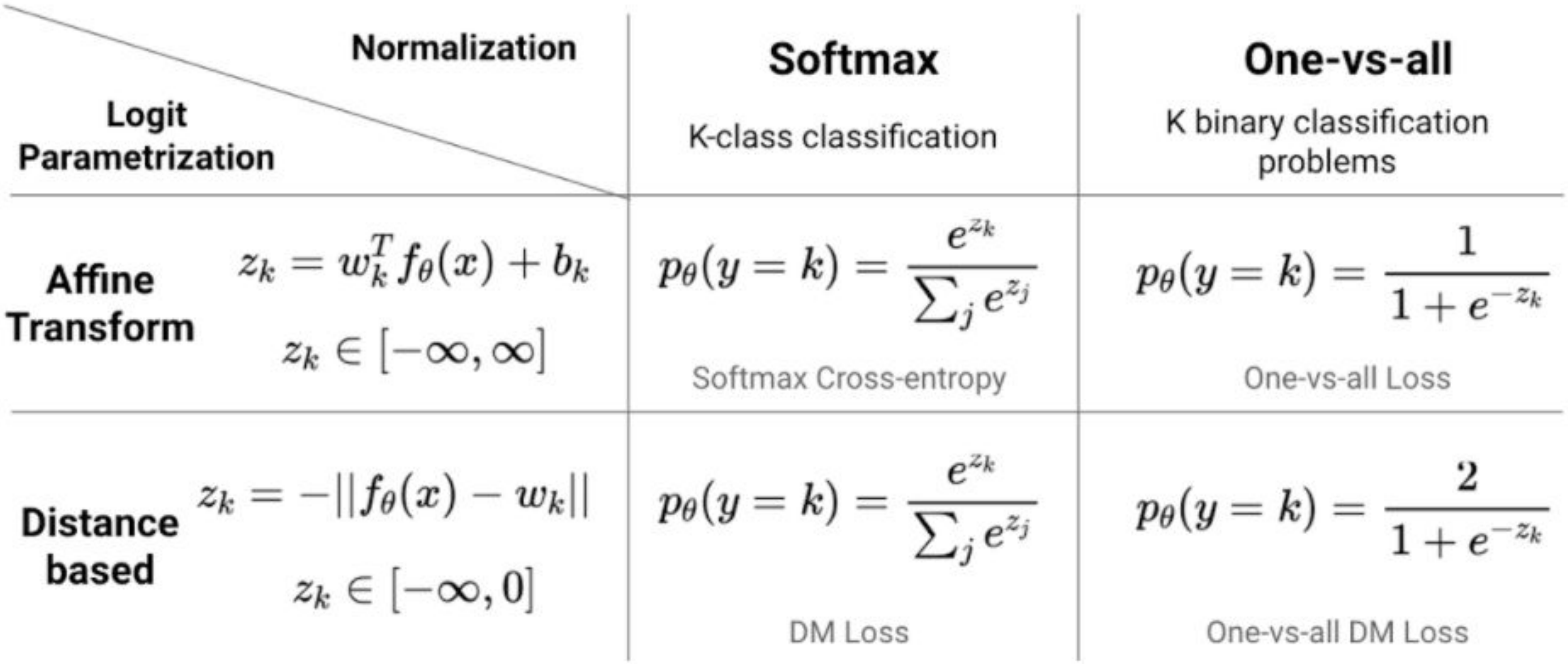
Detecting Out-of-Distribution Inputs

- **A Baseline for Detecting Misclassified & Out-of-Distribution Examples in Neural Networks** D. Hendrycks, K. Gimpel ICLR 2017
- **A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks** K. Lee, K. Lee, H. Lee, J. Shin NeurIPS 2018
- **Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks** S. Liang, Y. Li, R. Srikant ICLR 2018
- **Deep Anomaly Detection with Outlier Exposure** D. Hendrycks, M. Mazeika, T. Dietterich ICLR 2019

Questions?

Appendix

How should we parametrize probabilities?



Normalization Logit Parametrization		Softmax K-class classification	One-vs-all K binary classification problems
Affine Transform	$z_k = w_k^T f_\theta(x) + b_k$ $z_k \in [-\infty, \infty]$	$p_\theta(y = k) = \frac{e^{z_k}}{\sum_j e^{z_j}}$ Softmax Cross-entropy	$p_\theta(y = k) = \frac{1}{1 + e^{-z_k}}$ One-vs-all Loss
Distance based	$z_k = -\lvert\lvert f_\theta(x) - w_k \rvert\rvert$ $z_k \in [-\infty, 0]$	$p_\theta(y = k) = \frac{e^{z_k}}{\sum_j e^{z_j}}$ DM Loss	$p_\theta(y = k) = \frac{2}{1 + e^{-z_k}}$ One-vs-all DM Loss

Image source: Padhy et al. 2020 "Revisiting One-vs-All Classifiers for Predictive Uncertainty and Out-of-Distribution Detection in Neural Networks"