

Practical Introduction to Uncertainty in Deep Learning

Disclaimer: Work in progress. Portions of these written materials are incomplete.

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**.

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

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

(Independent and Identically Distributed)

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

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

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

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

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

Neural networks do not generalize under covariate shift

- **Accuracy drops** with increasing shift on Imagenet-C
- But do the models know that they are less accurate?

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

- **Accuracy drops** with increasing shift on Imagenet-C
- **Quality of uncertainty degrades with shift**
-> “overconfident mistakes”

Models assign high confidence predictions to OOD inputs

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

Models assign high confidence predictions to OOD inputs

Models assign high confidence predictions to OOD inputs

Primer on Uncertainty & Robustness

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)

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*

Sources of uncertainty: *Data uncertainty*

- 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*

How do we measure the quality of uncertainty?

$$\text{Calibration Error} = |\text{Confidence} - \text{Accuracy}|$$

*predicted probability
of correctness*

*observed frequency
of correctness*

How do we measure the quality of uncertainty?

$$\text{Calibration Error} = |\text{Confidence} - \text{Accuracy}|$$

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?

$$\text{Calibration Error} = |\text{Confidence} - \text{Accuracy}|$$

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](#)]:

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

- 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](#)]:

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



How do we measure the quality of uncertainty?

Expected Calibration Error [Naeini+ 2015]:

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

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).

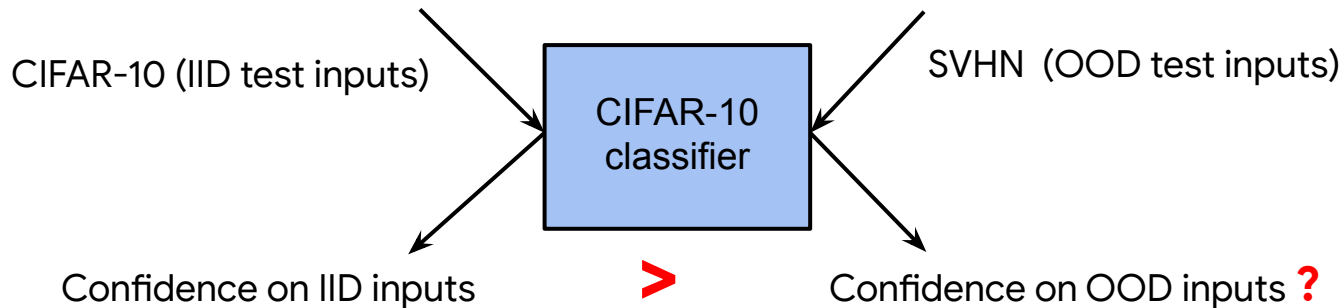
$$\text{BS} = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} [p(y|\mathbf{x}_n, \theta) - \delta(y - y_n)]^2$$

- Can be numerically unstable to optimize.

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 $p(y|x)$



Fundamentals to Uncertainty & Robustness Methods

Neural Networks with SGD

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

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

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

Neural Networks with SGD

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

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

↑
Data uncertainty

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

Neural Networks with SGD

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

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

How do we get uncertainty?

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

Probabilistic Machine Learning

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

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

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

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

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

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

Bayesian Neural Networks

Bayesian neural nets specify a distribution over neural network predictions.

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

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

Approximating the posterior

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

Approximating the posterior

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

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

Under Gaussian likelihood + prior and
in the limit of infinite basis functions (e.g. hidden units) \rightarrow 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?

See [[Rasmussen & Williams 2006](#)]

Gaussian Processes

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

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

The result is a flexible distribution over functions

- Specified now by a covariance function over examples $K(X, X)$
- Get a posterior on functions conditioned on data

$\mathbf{f}_*|X, \mathbf{y}, X_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*)),$ where

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}[\mathbf{f}_*|X, \mathbf{y}, X_*] = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} \mathbf{y},$$

$$\text{cov}(\mathbf{f}_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} K(X, X_*)$$

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
- 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.

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]

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.

Caveat: Dataset shift...

[Guo+ 2017](#) “On calibration of modern neural networks”

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

Simple Baseline: Monte Carlo Dropout

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

Deep Ensembles work surprisingly well in practice

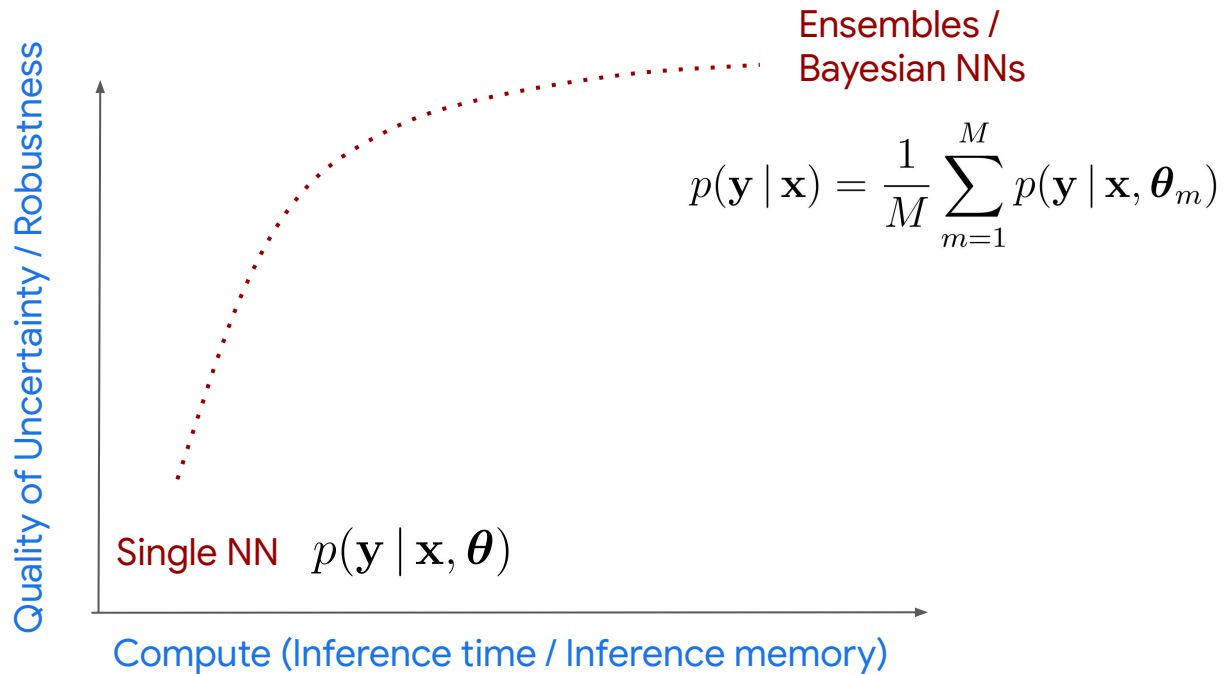
Deep Ensembles are consistently among the best performing methods, especially under dataset shift

Simple Baseline: Bootstrap

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

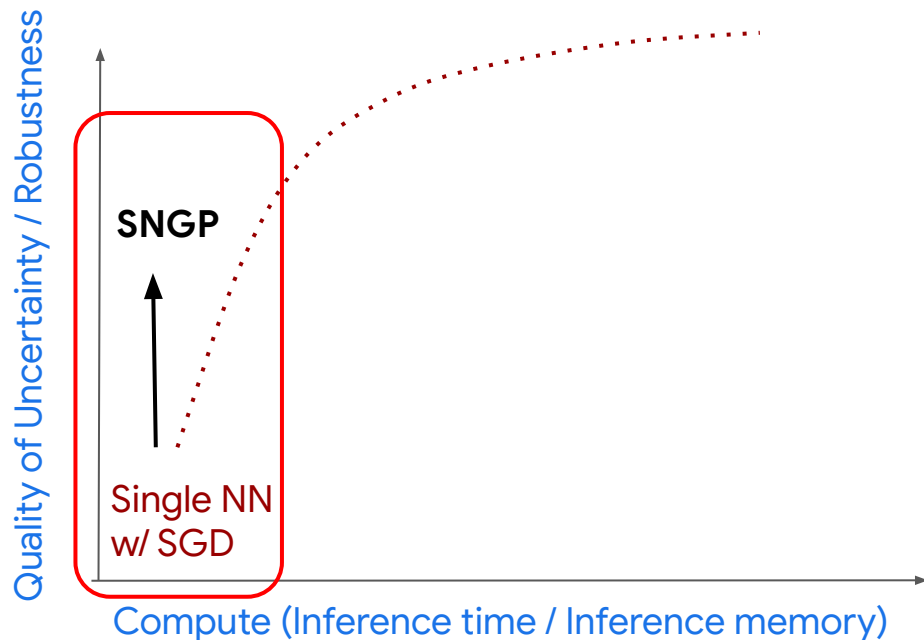
- Resample the dataset with replacement and retrain
- Each example gets a different weight under each model

Recent highlights



Adding *distance-awareness* using **Spectral-normalized Neural Gaussian Process (SNGP)**

High uncertainty
(low confidence)



Low uncertainty
(high confidence)

SNGP assigns lower confidence predictions to inputs far away from the training data

Imposing distance awareness

Data augmentation is effective for enforcing invariant predictions under shift.

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

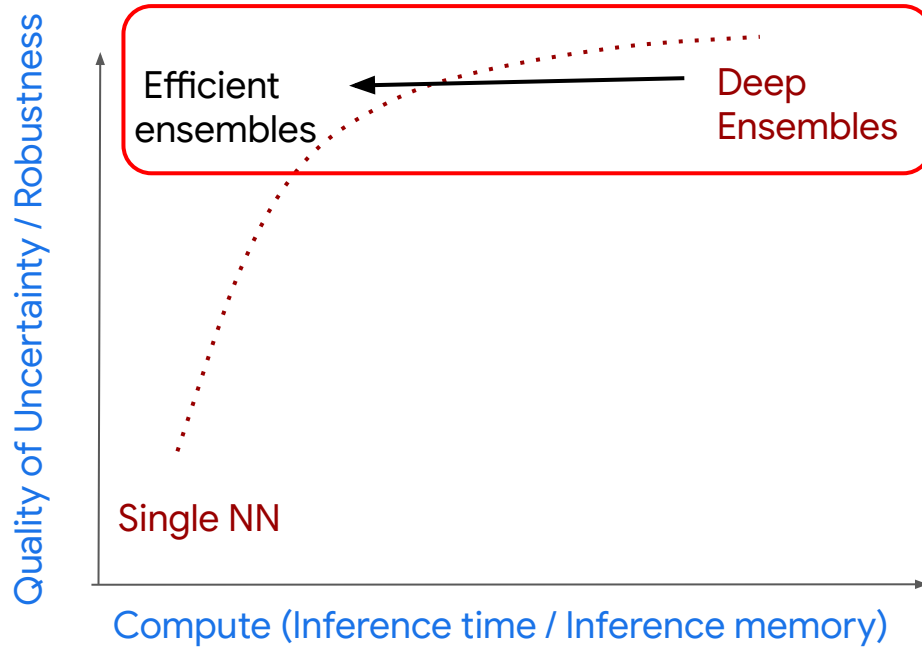
Spectral-normalized Neural Gaussian process

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

See also [[van Amersfoort+ 2020](#)].

[[Liu+ 2020](#)]

Efficient ensembles lower inference memory and/or inference time



Multi-Input Multi-Output (MIMO)

$(x_1, y_1), (x_2, y_2), (x_3, y_3)$ sampled independently during training

$x_1 = x_2 = x_3$ during evaluation training

Data augmentation, e.g. AugMix

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

AugMix improves accuracy & calibration under shift

Data augmentation can provide complementary benefits to ensembling.

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.