Trustworthy AI Systems

-- Robustness of AI

Instructor: Guangjing Wang

guangjingwang@usf.edu

Last Lecture

- Accountability
- Detecting AI-generated Content
- Watermarking Techniques
- Evading Watermarking-based Detection

This Lecture

- Uncertainty and Robustness

- Source of Uncertainty

- Measure the Quality of Uncertainty

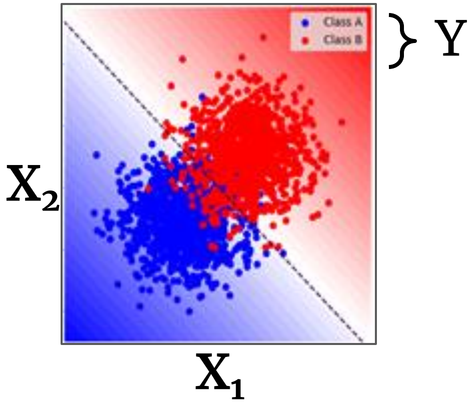
- Reduce Uncertainty and Enhance Robustness

What do we mean by Uncertainty?

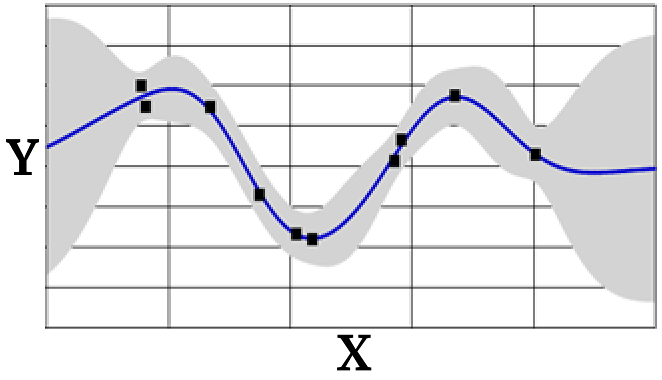
Return a distribution over predictions rather than a single prediction.

- ***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.***



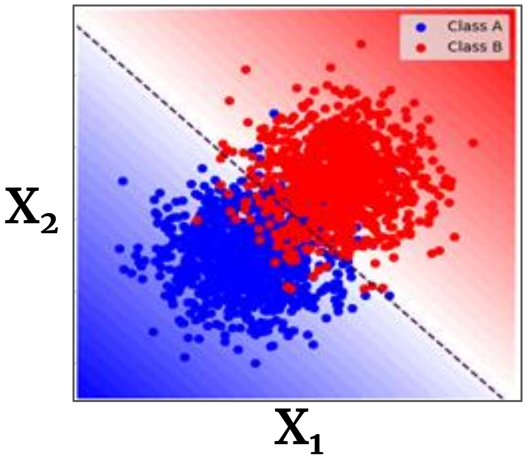
$$p(\mathbf{y}|\mathbf{x})$$



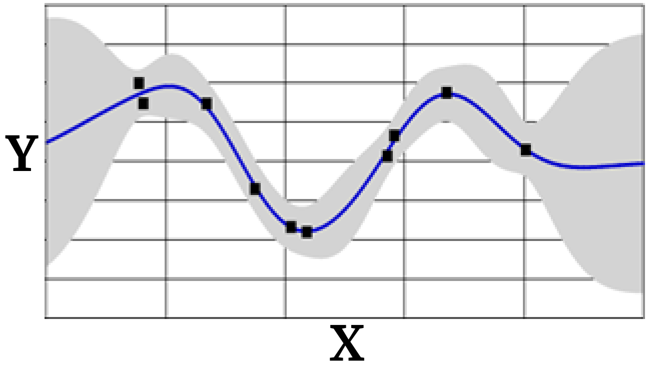
What do we mean by Out-of-Distribution Robustness?

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

(*Independent and Identically Distributed*)



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



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

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

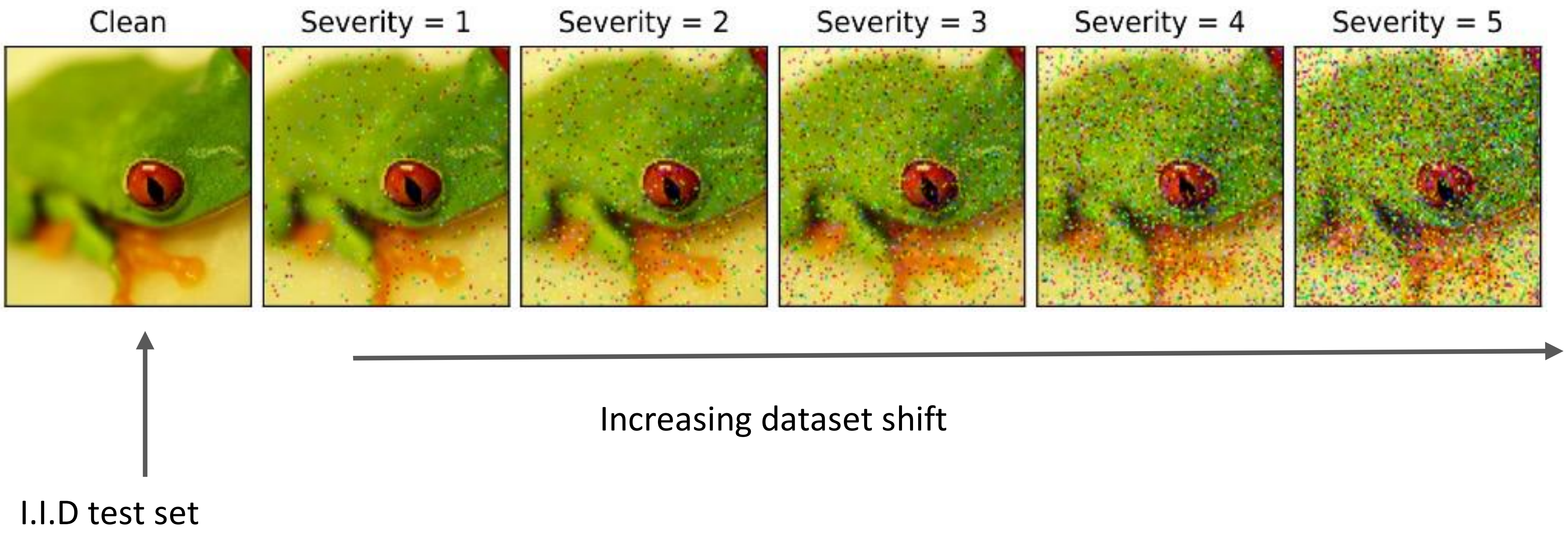
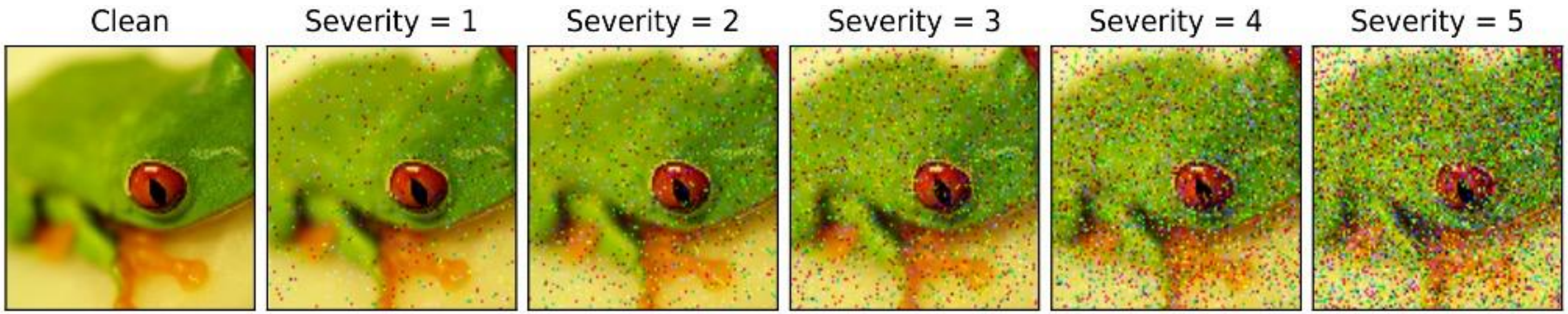


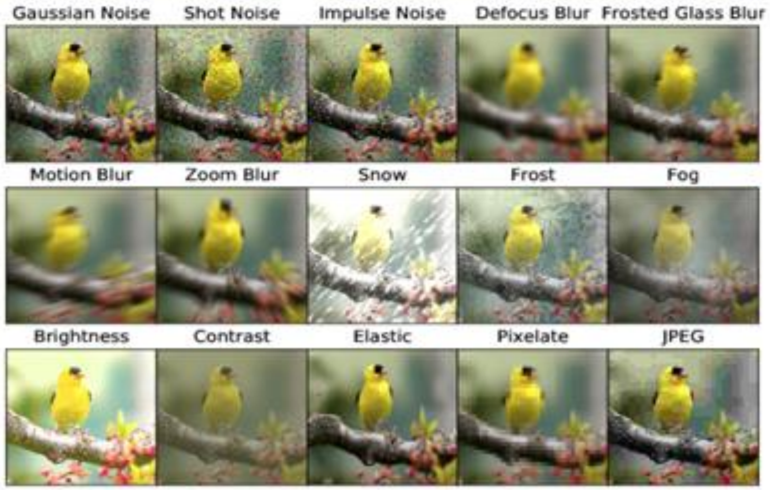
Image source: Benchmarking Neural Network Robustness to Common Corruptions and Perturbations, Hendrycks & Dietterich, 2019.

ImageNet-C: Varying Intensity for Dataset Shift



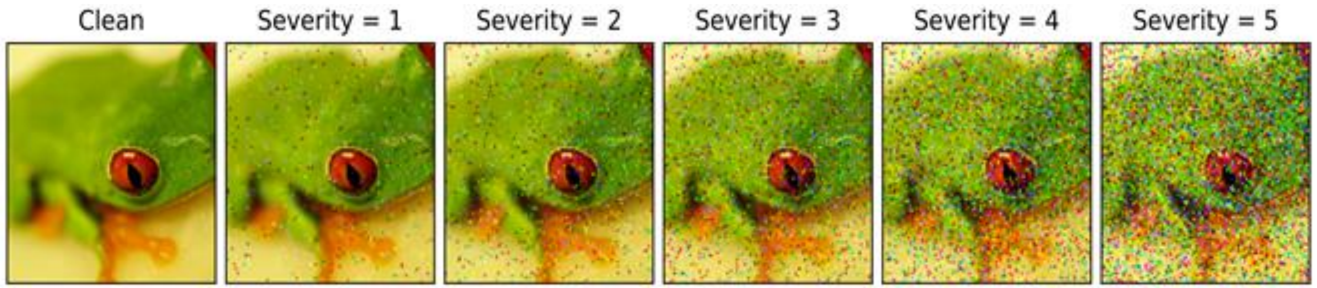
Increasing dataset shift

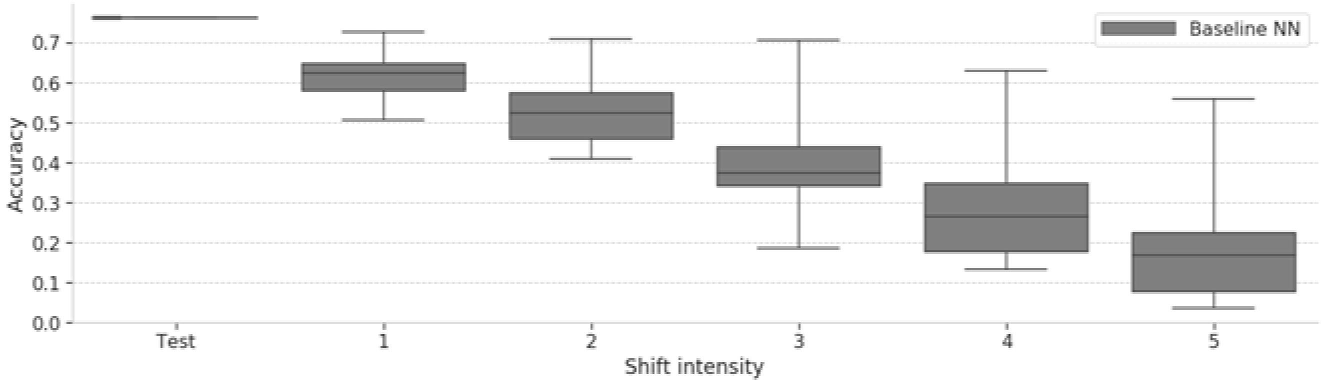
I.I.D test set



Neural networks do not generalize under covariate shift

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



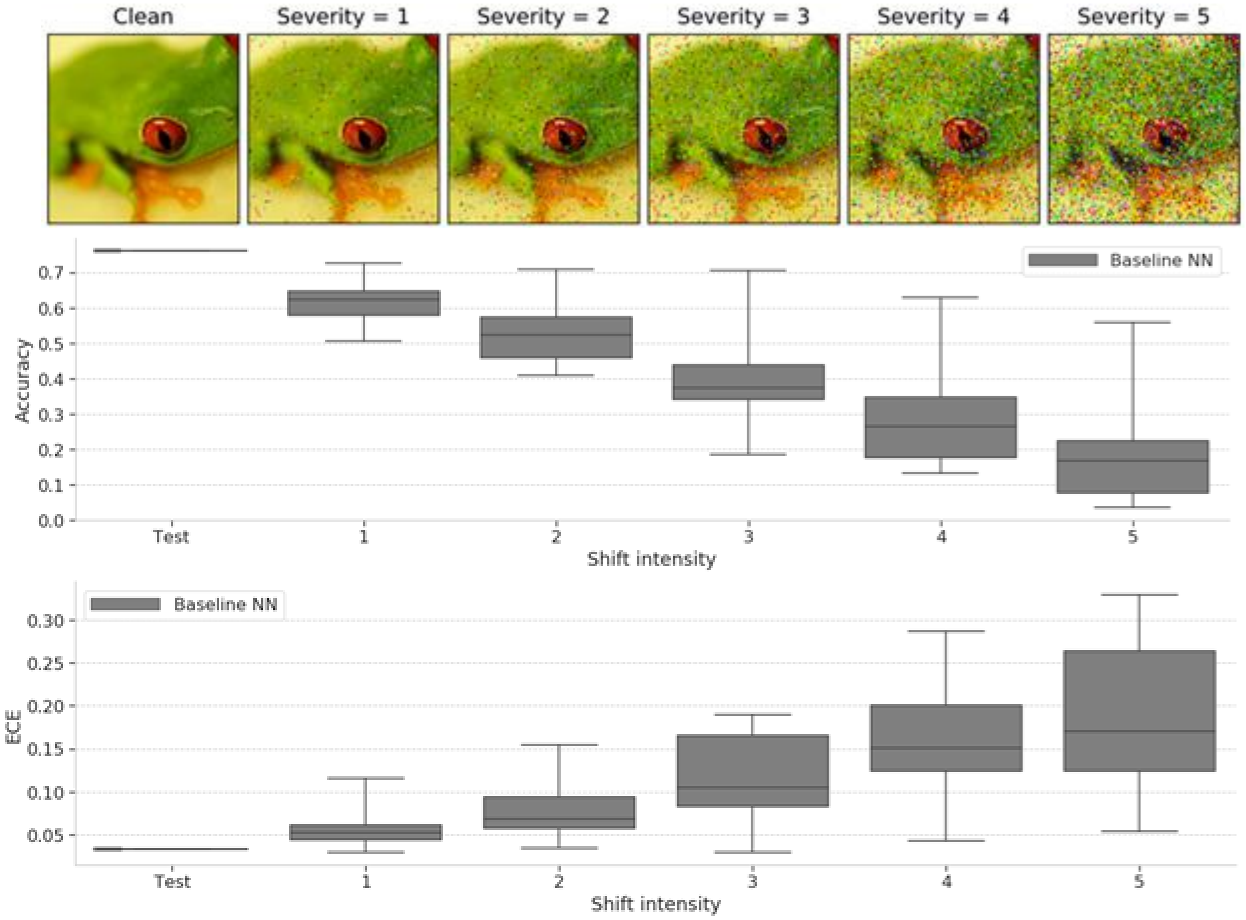


- But do the models know that they are less accurate?

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

Example images where model assigns >99.5% confidence.

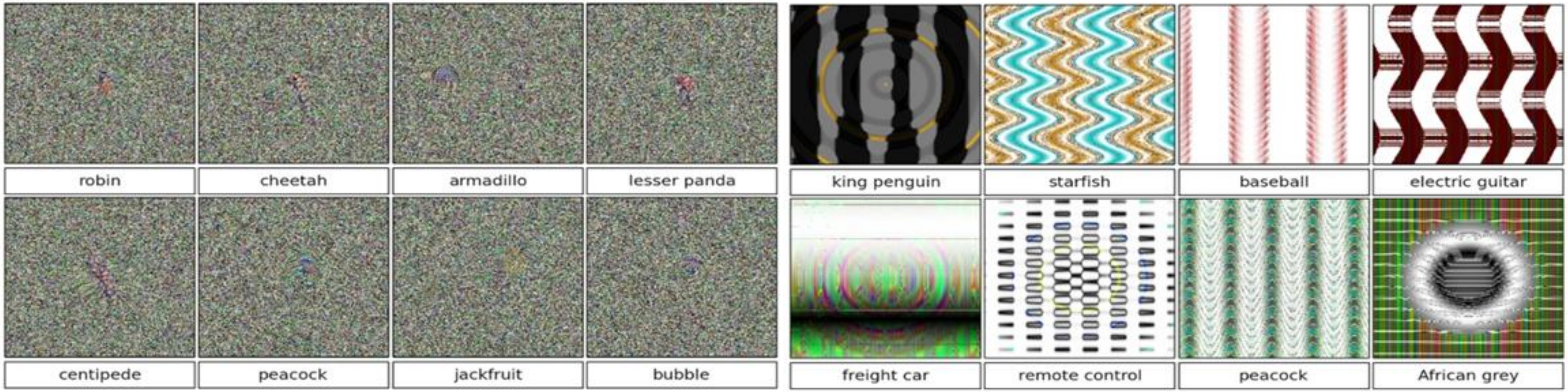
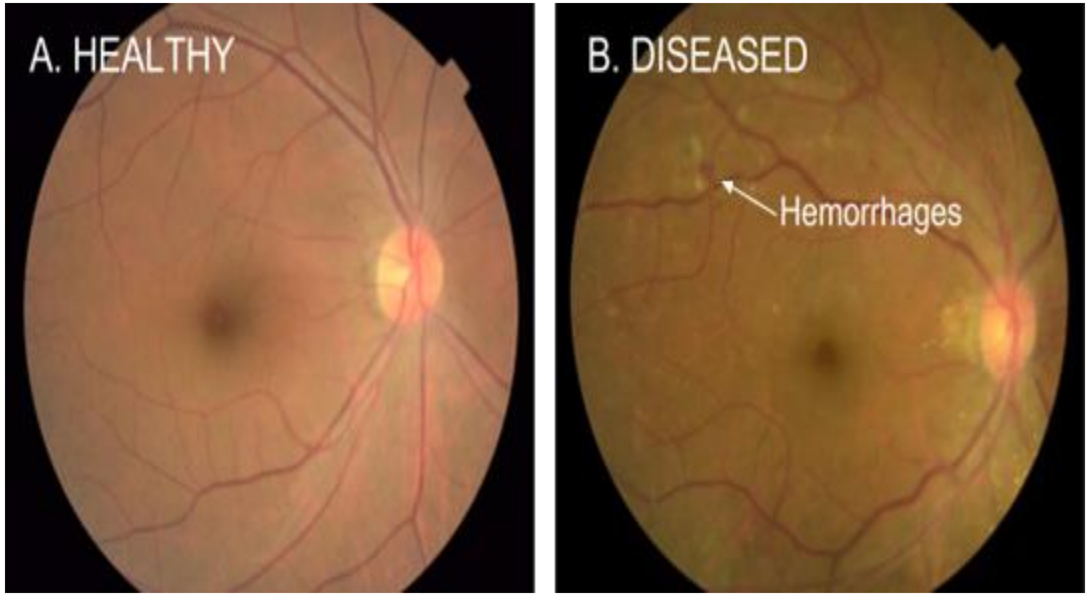


Image source: "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images" [Nguyen et al. 2014]

Healthcare (1)



Diabetic retinopathy detection from fundus images Gulshan et al, 2016

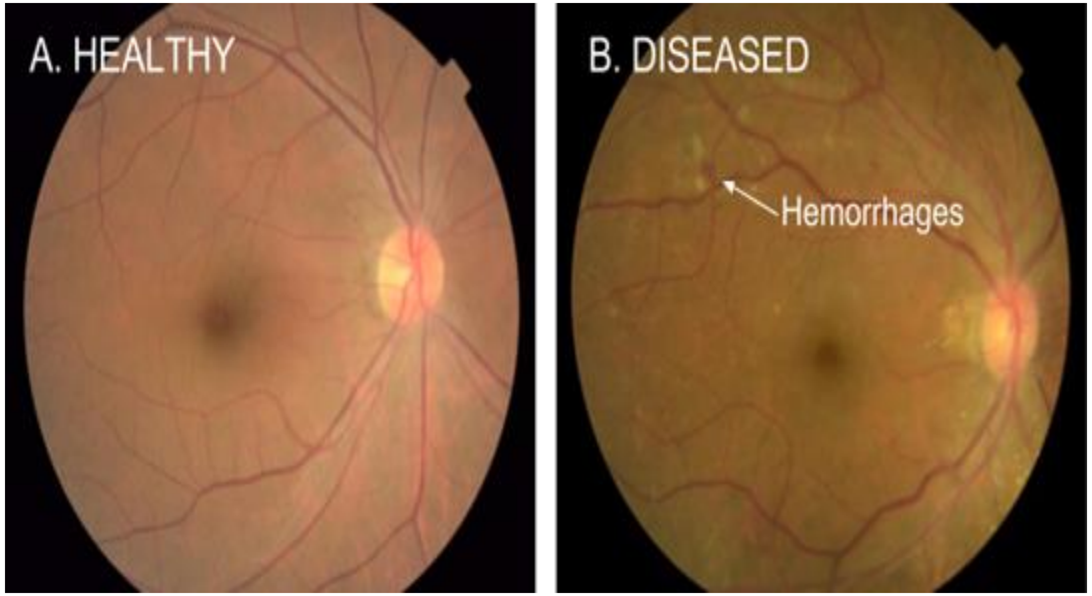
True label

Predicted label		Healthy	Diseased
	Healthy	0	10
	Diseased	1	0

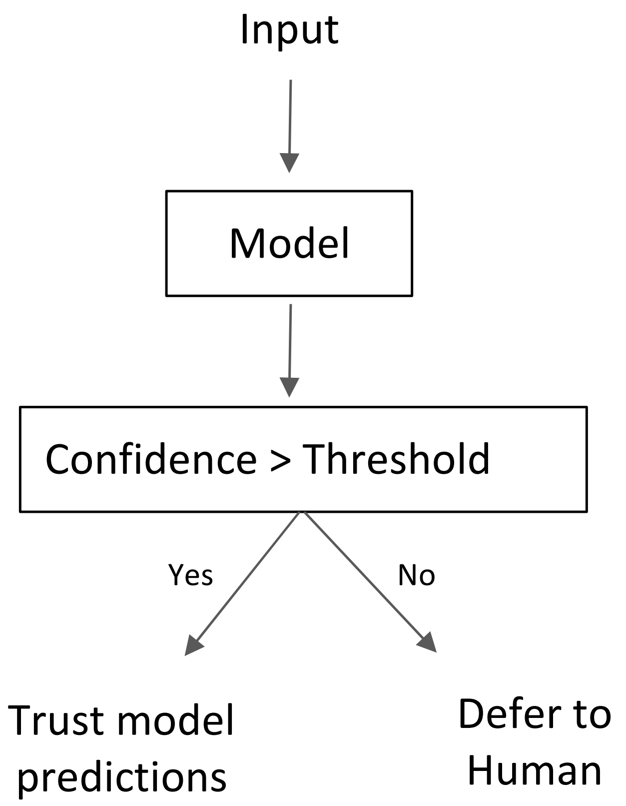
Cost-sensitive decision making

Healthcare (2)

- 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]



Self-driving Cars

Dataset shift:

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

Image credit: Sun et al, Waymo Open Dataset

Open Set Recognition

- Example: Classification of genomic sequences

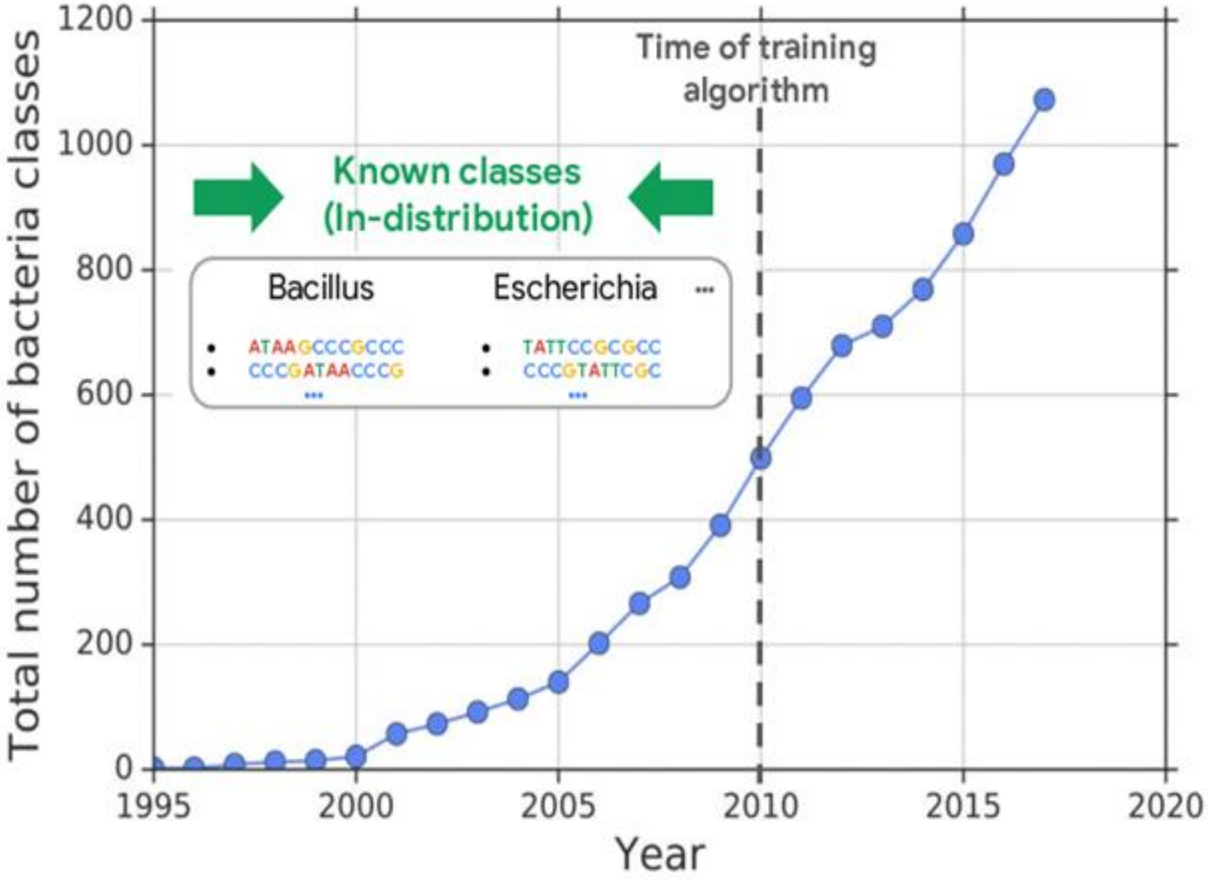
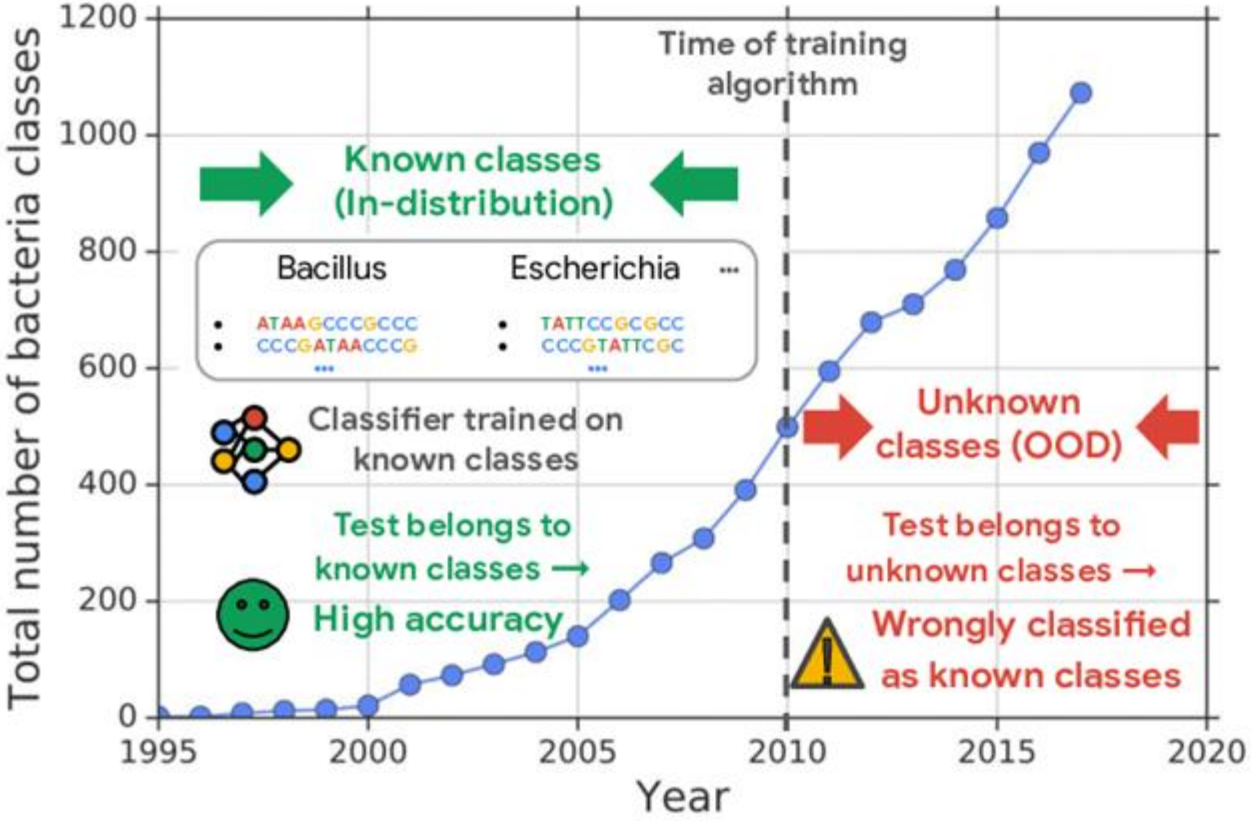


Image source: https://ai.googleblog.com/2019/12/improving-out-of-distribution-detection.html

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



Conversational Dialog Systems

- 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"

Uncertainty in Other Areas

Safety

Decision making

Active learning
Lifelong learning

Open-set recognition

Uncertainty & Out-of-Distribution Robustness

Reinforcement learning

Graceful failure

Trustworthy ML

Bayesian optimization

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

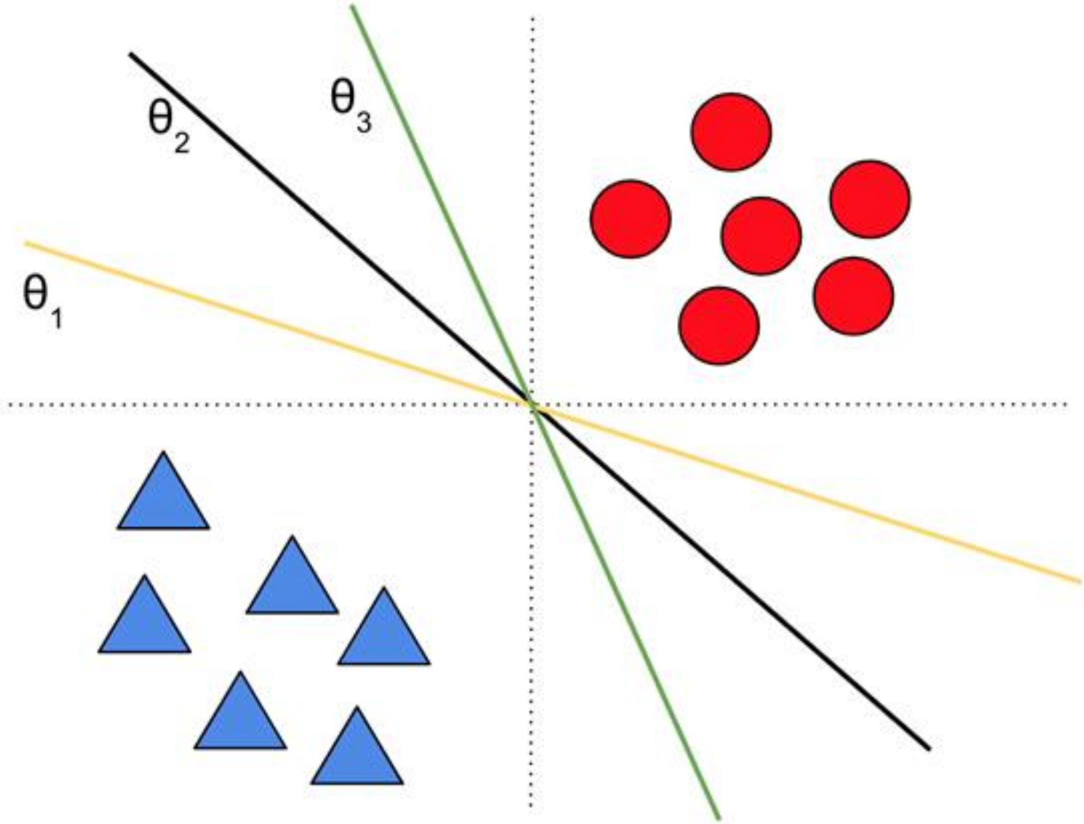
This Lecture

- Uncertainty and Robustness
- Source of Uncertainty
- Measure the Quality of Uncertainty
- Reduce Uncertainty and Enhance Robustness

Sources of uncertainty: Model uncertainty

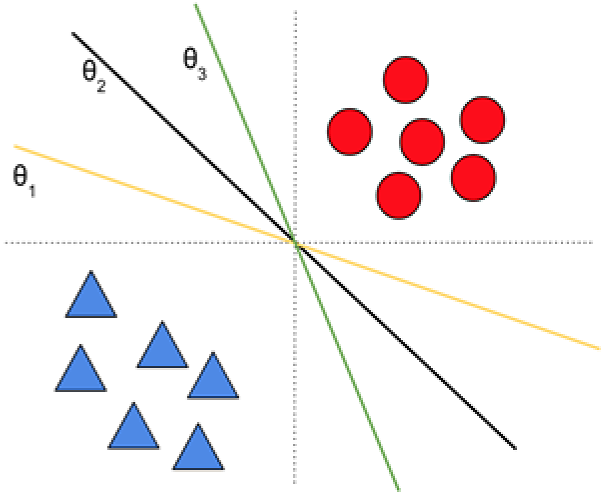
- Many models can fit the training data well
- Also known as ***epistemic uncertainty***
- Model uncertainty is "**reducible**"
  - Vanishes in the limit of infinite data

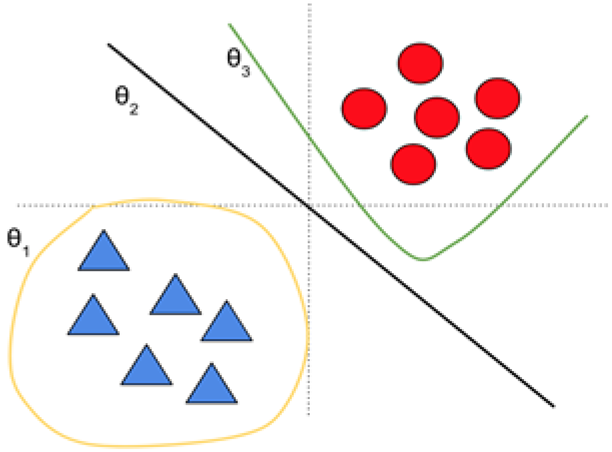
    (subject to model identifiability)



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

- 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

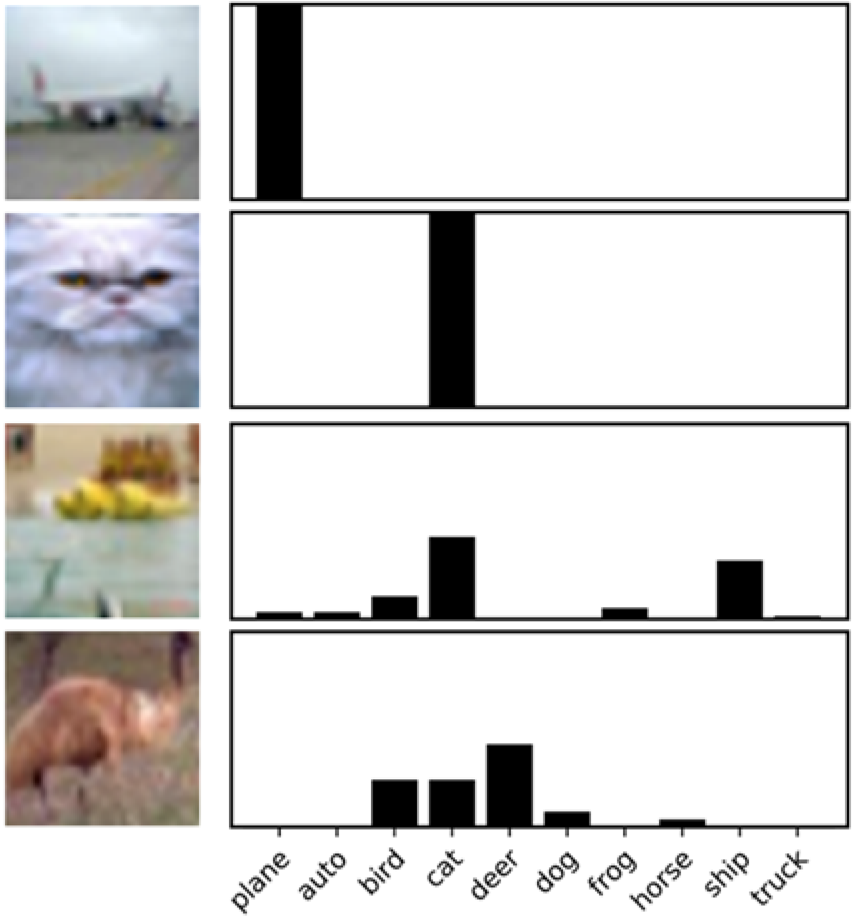


Image source: Battleday et al. 2019 "Improving machine classification using human uncertainty measurements"

This Lecture

- Uncertainty and Robustness
- Source of Uncertainty
- Measure the Quality of Uncertainty
- Reduce Uncertainty and Enhance Robustness

How do we measure the quality of uncertainty?

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

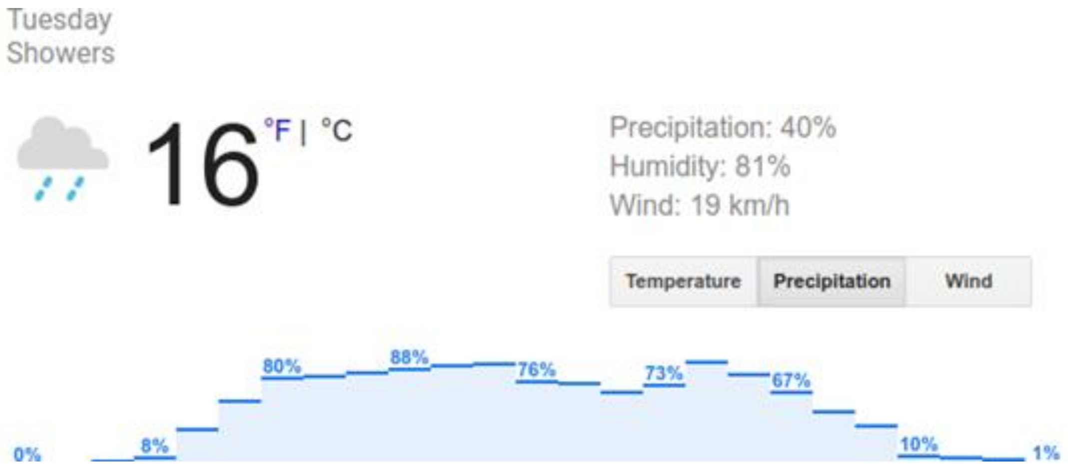
predicted probability of correctness

observed frequency of correctness

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?

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

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]:

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

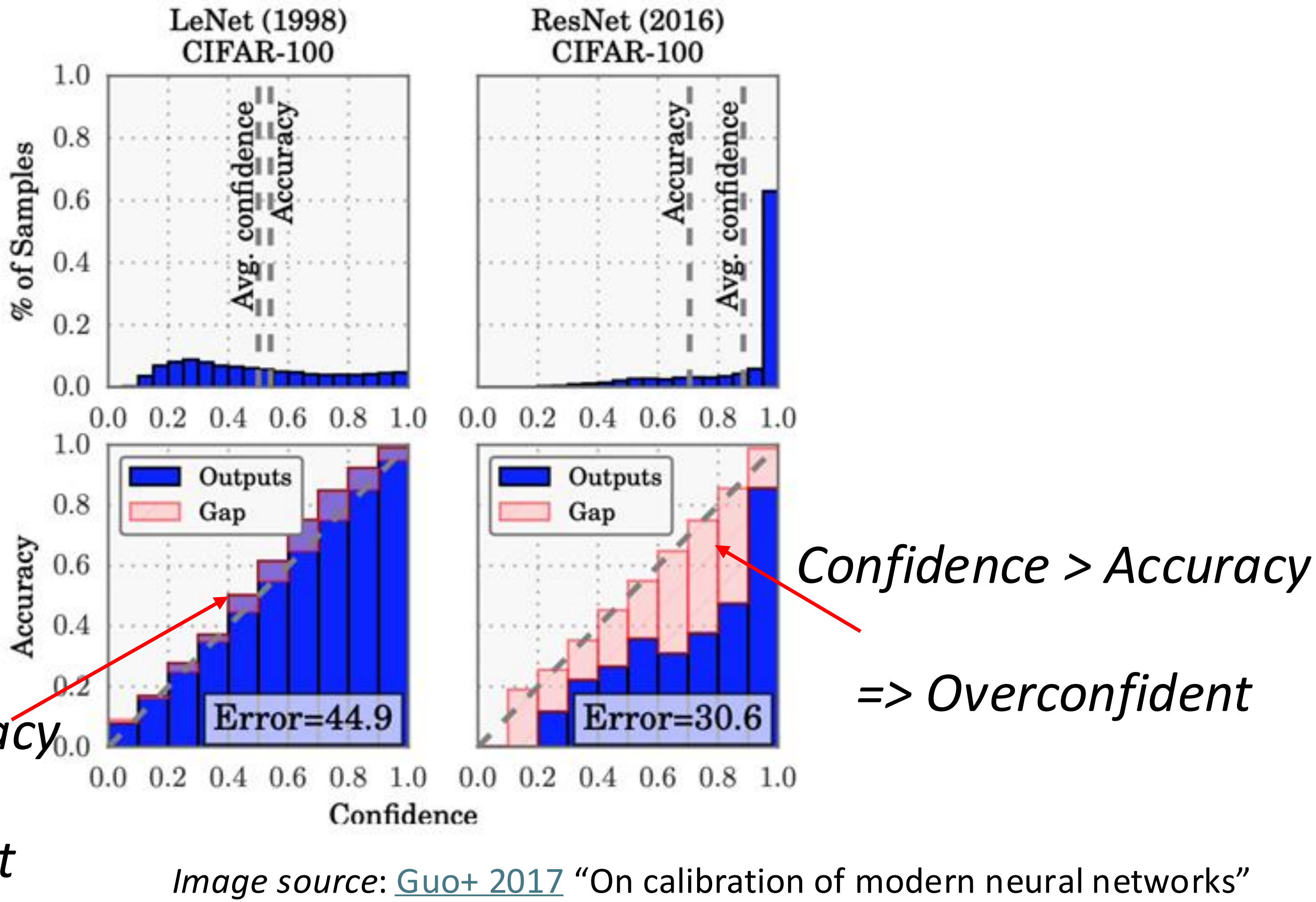


Image source: Guo+ 2017 “On calibration of modern neural networks”

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**.

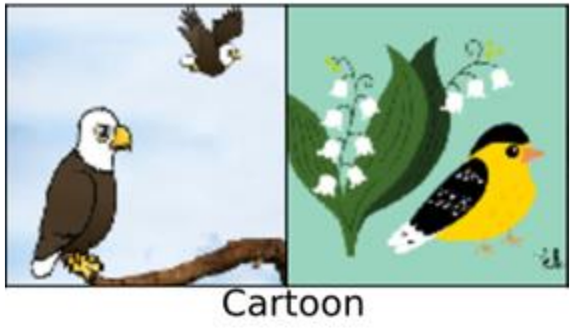
Predicting class frequency p(y=1) = 0.3 for all the inputs achieves perfect calibration.

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 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)

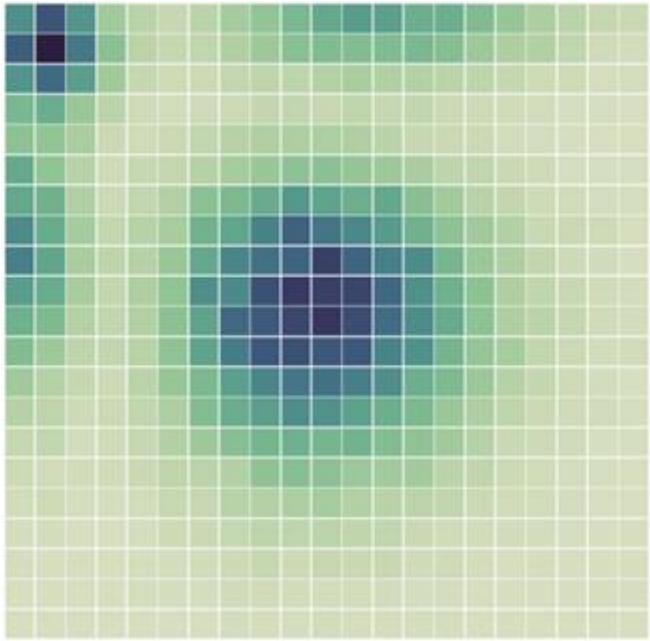
This Lecture

- Uncertainty and Robustness
- Source of Uncertainty
- Measure the Quality of Uncertainty
- Reduce Uncertainty and Enhance Robustness

Neural Networks with SGD (1)

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

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



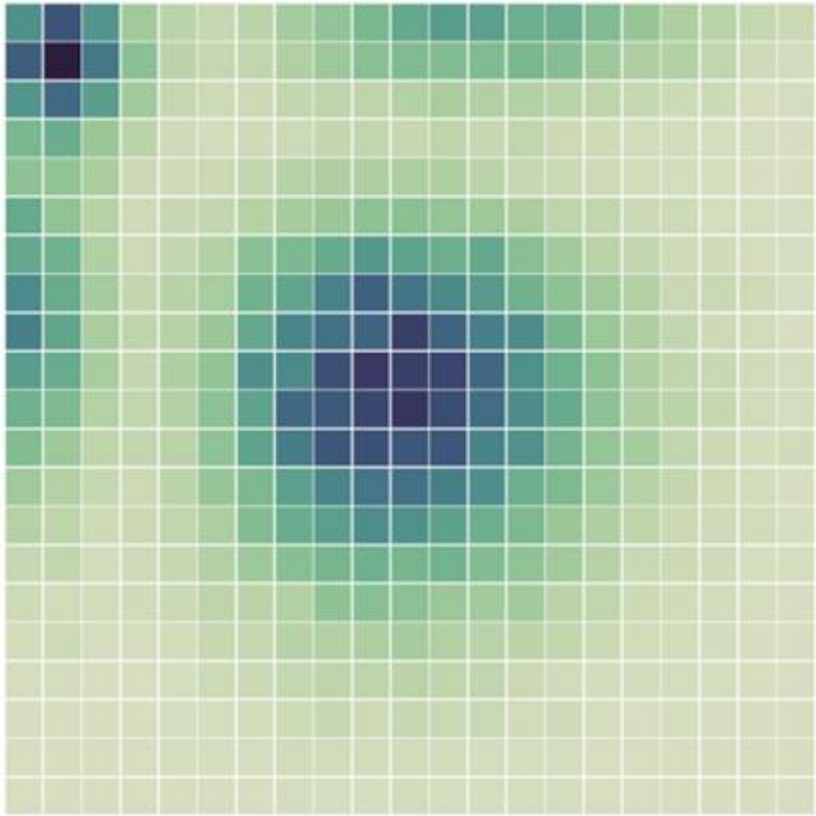
Special case: softmax cross entropy with L2 regularization. Optimize with SGD!

Neural Networks with SGD (2)

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

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

Data uncertainty

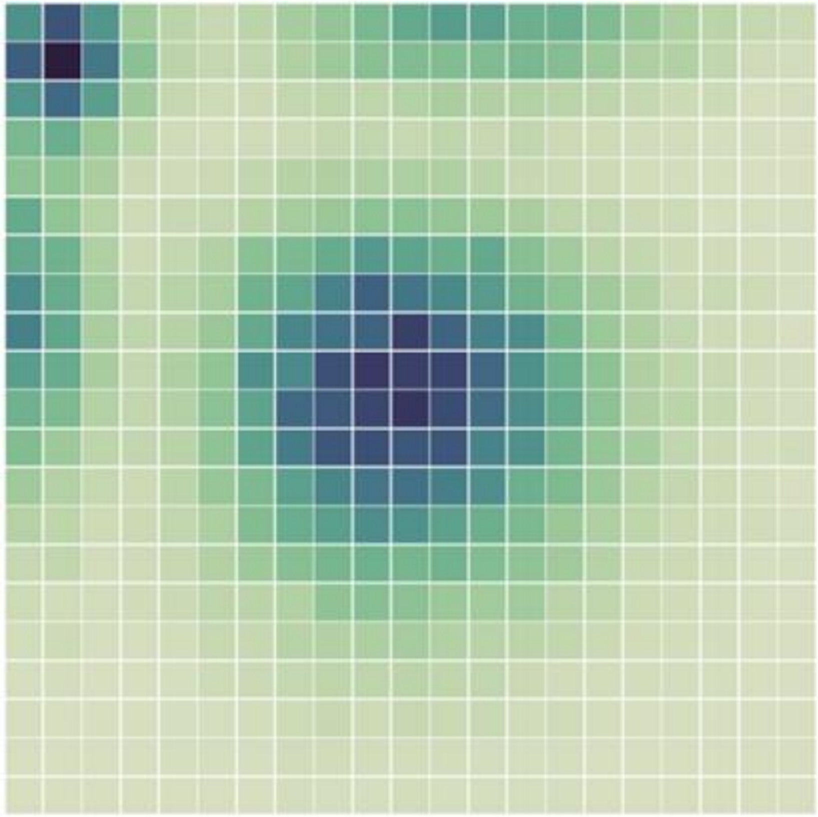


Special case: softmax cross entropy with L2 regularization. Optimize with SGD!

Neural Networks with SGD (3)

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

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



How do we get uncertainty?

- Probabilistic approach
  - Estimate a full distribution for
- Intuitive approach: Ensembling
  - Obtain multiple good settings for $\boldsymbol{\theta}^*$

Probabilistic Machine Learning

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

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

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

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

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

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

Bayesian Neural Networks

Bayesian neural nets specify a distribution over neural network predictions.

This is done by specifying a distribution over neural network weights .

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

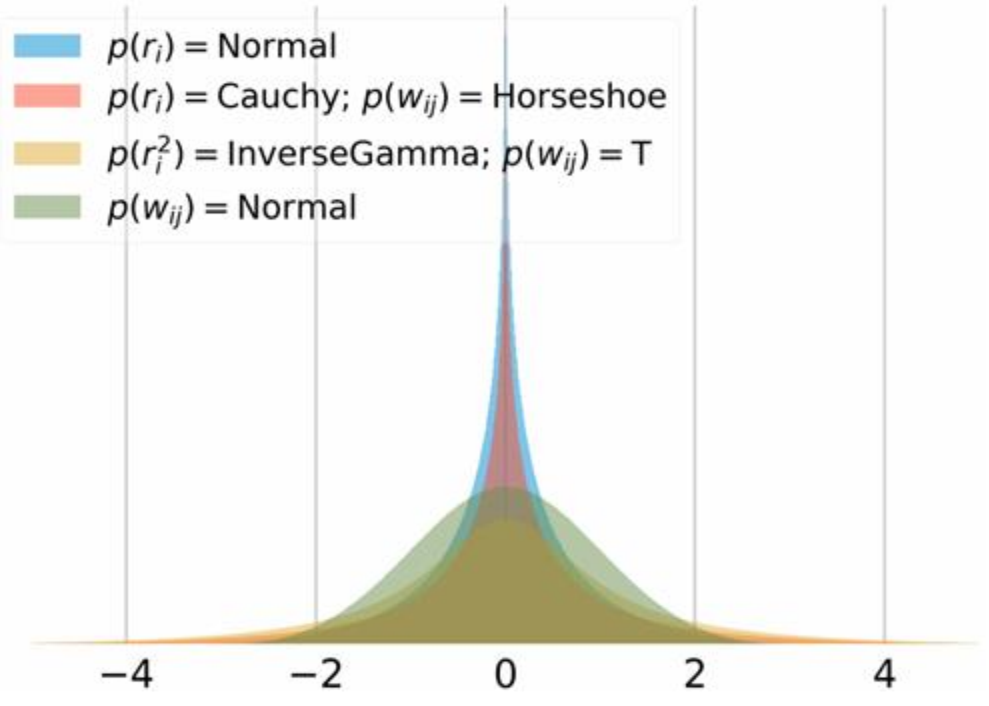
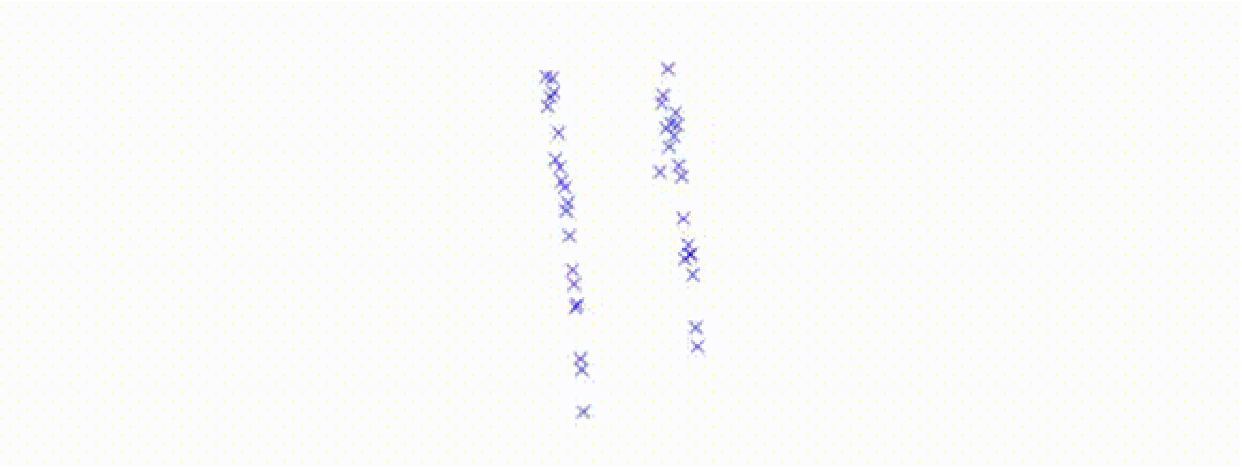
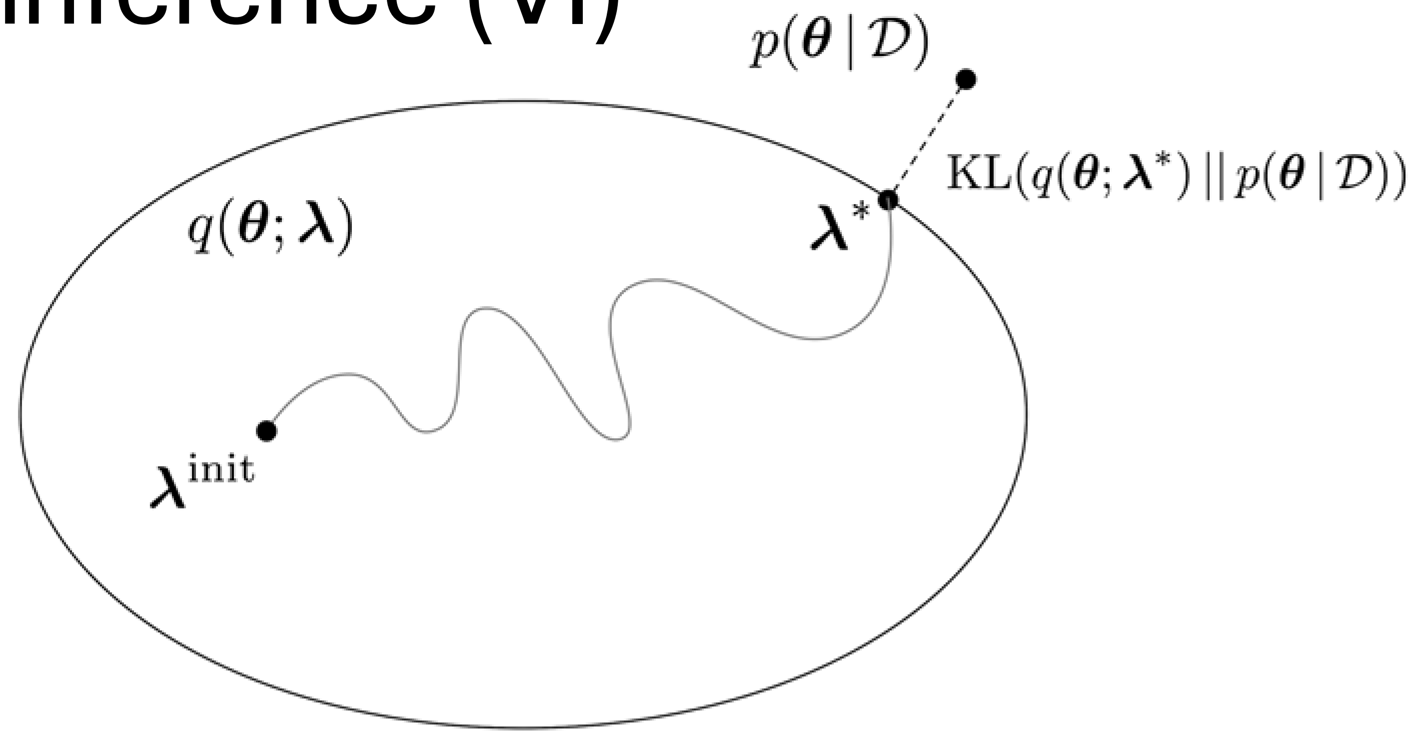


Image source: Dusenberry+ 2020



Variational inference (VI)

D is a collection of possible models



- 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 .

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]

Ensemble Learning

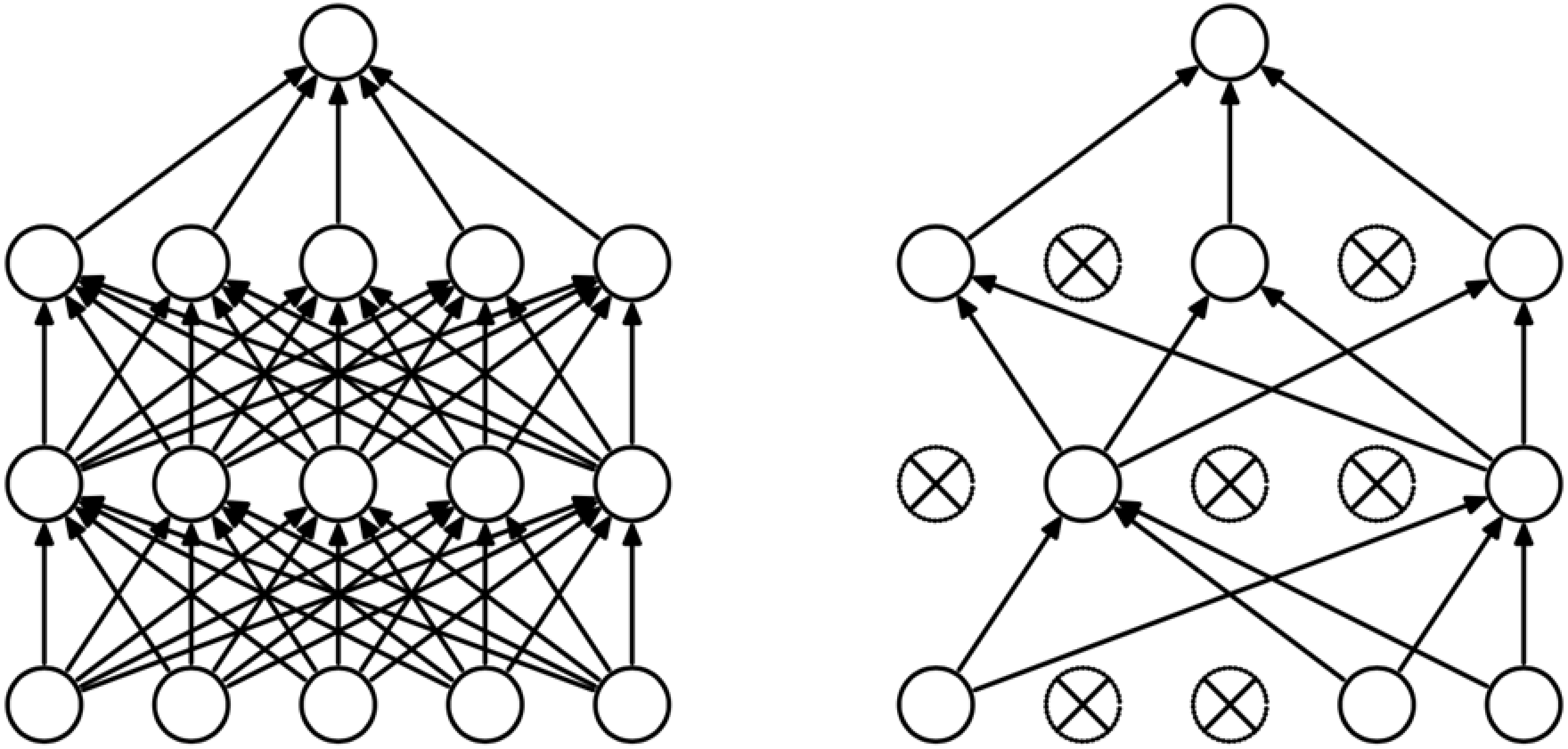
- A prior distribution often involves the complication of approximate inference.
- *Ensemble learning* offers an alternative strategy to aggregate the predictions over a collection of models.
- Often winner of competitions!
- There are two considerations: the collection of models to ensemble; and the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

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

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

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

Monte Carlo Dropout



(a) Standard Neural Net

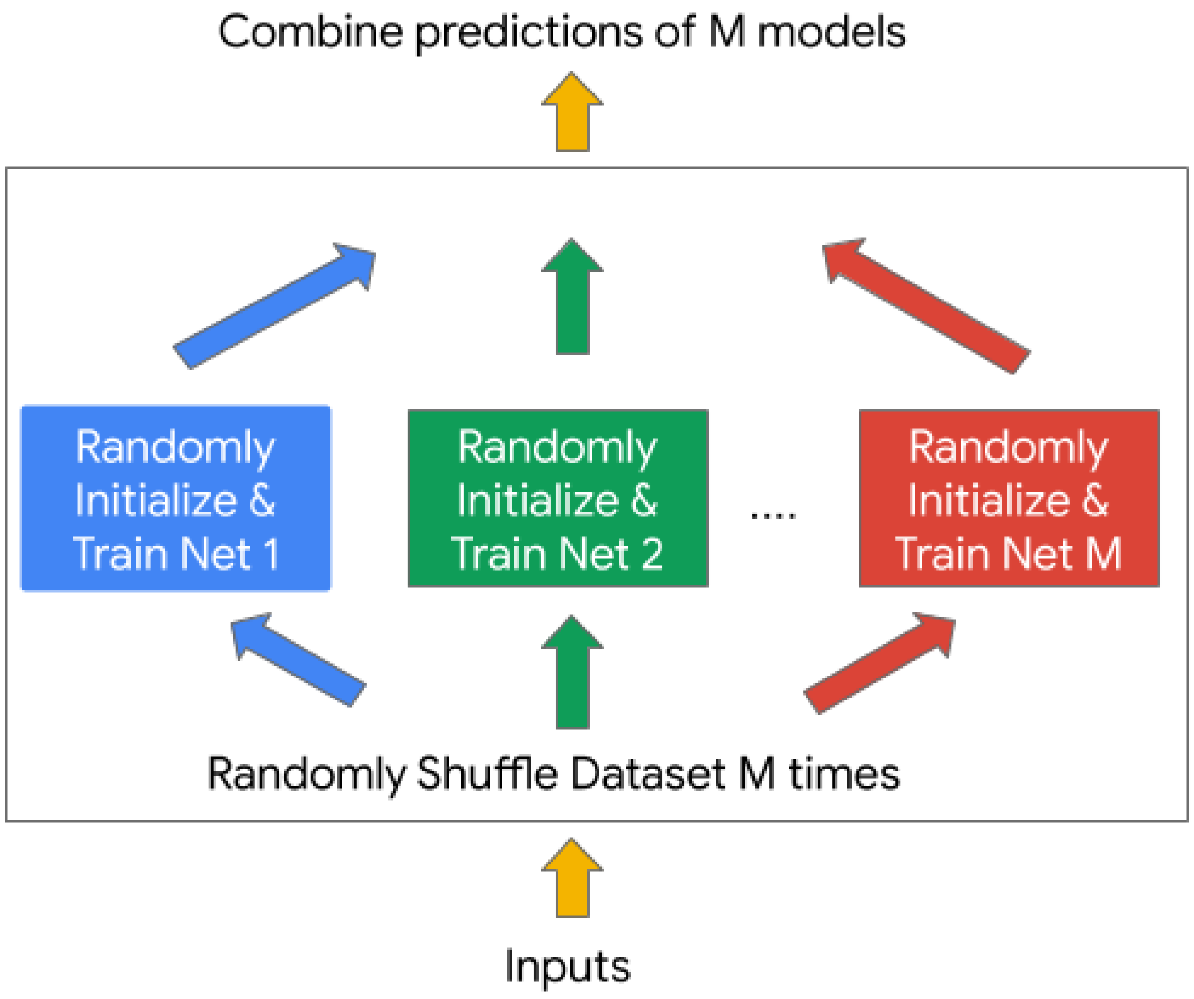
(b) After applying dropout.

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

Deep Ensembles

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

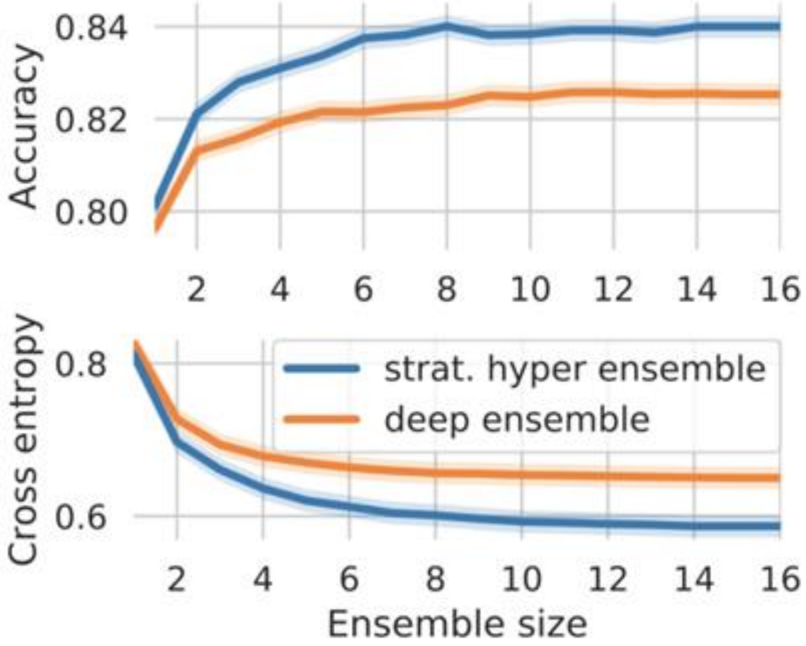
- A well known trick for getting better accuracy and Kaggle scores
- We rely on the fact that the loss landscape is non-convex to land at different solutions
    - Rely on different initializations and SGD noise



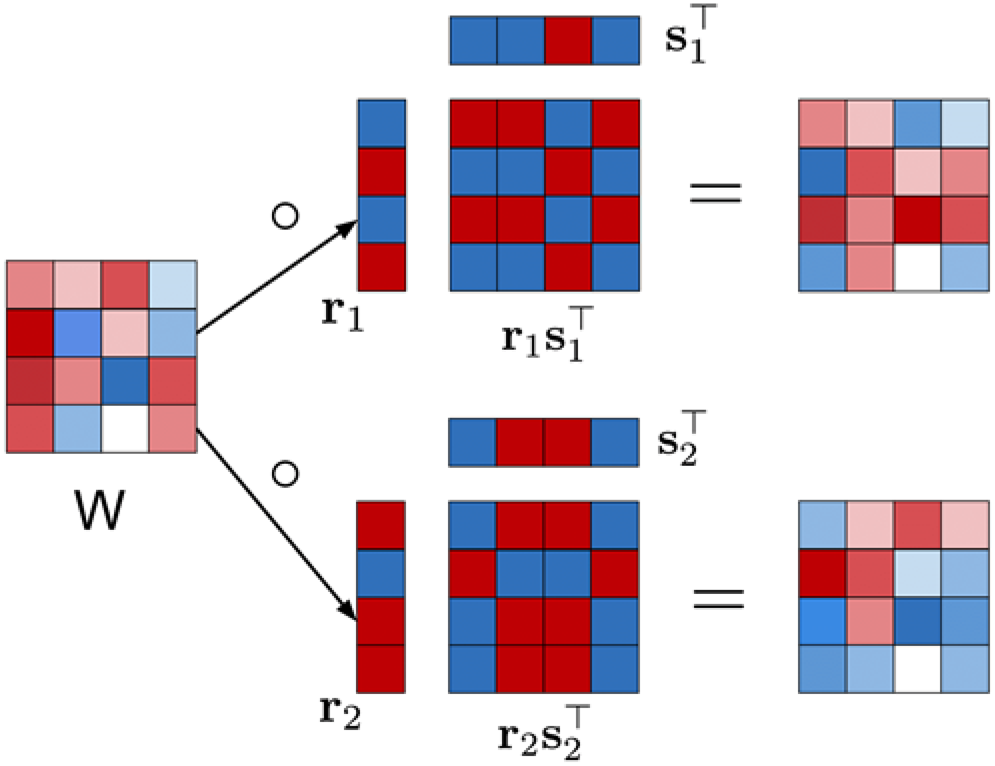
Hyperparameter Ensembles

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

- Run random search to generate a set of models.
    - Include random seed as part of the search space.
- Greedily select the K models to pool.



Efficient Ensembles by Sharing Parameters (1)



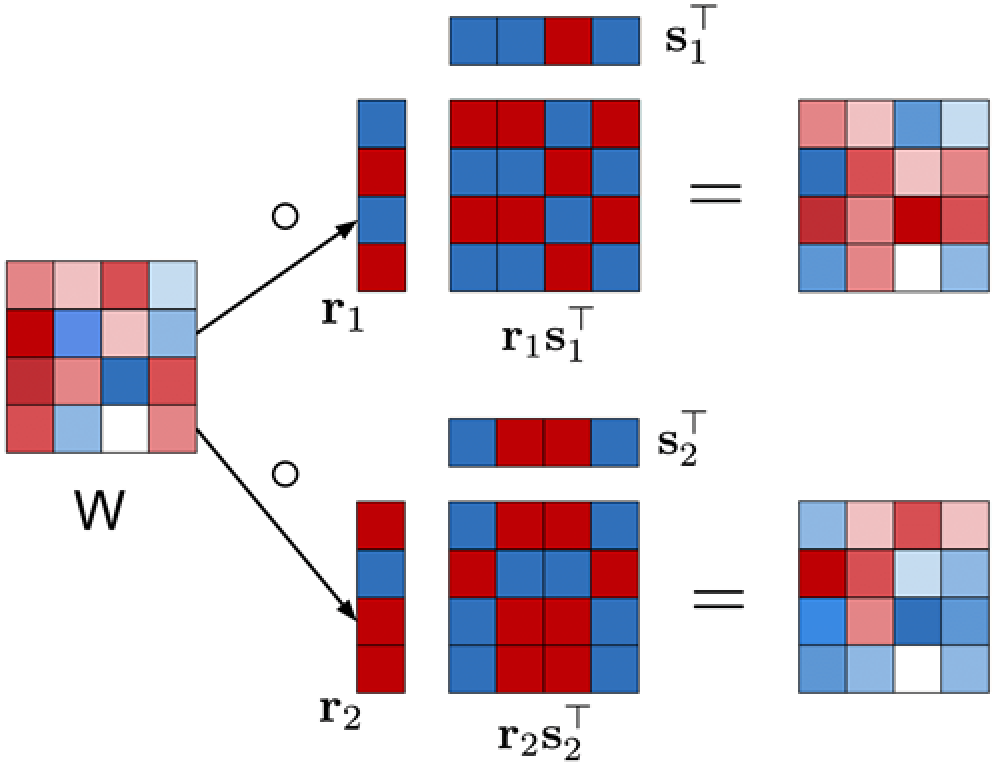
Parameterize each weight matrix as a new weight matrix ***W*** multiplied by the outer product of two vectors ***r*** and **s**.

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

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

Known as **BatchEnsemble**.

Efficient Ensembles by Sharing Parameters (2)



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*.

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

Simple Baseline: Recalibration

For classification, modify softmax probabilities post-hoc.

Temperature Scaling.

1. Parameterize output layer with scalar T.

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

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

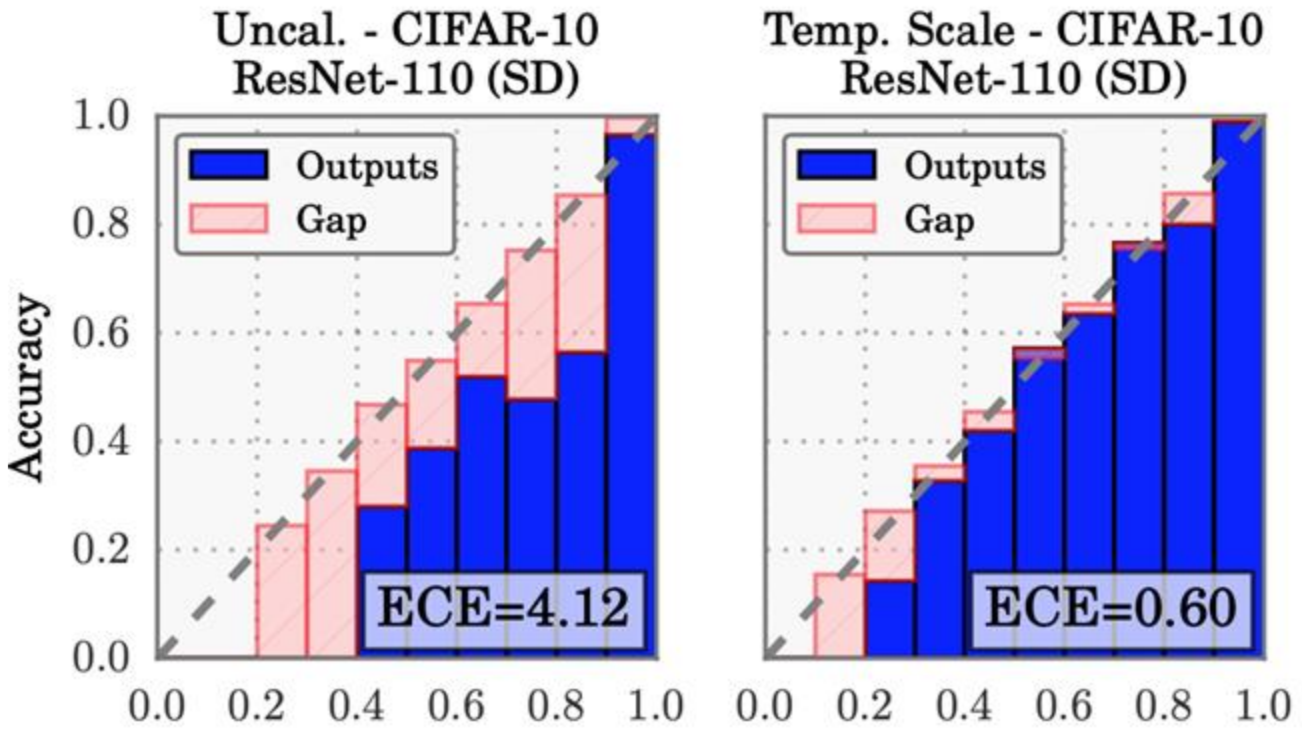


Image source: Guo+ 2017 "On calibration of modern neural networks"

Caveat: Dataset shift...

Uncertainty Baselines

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

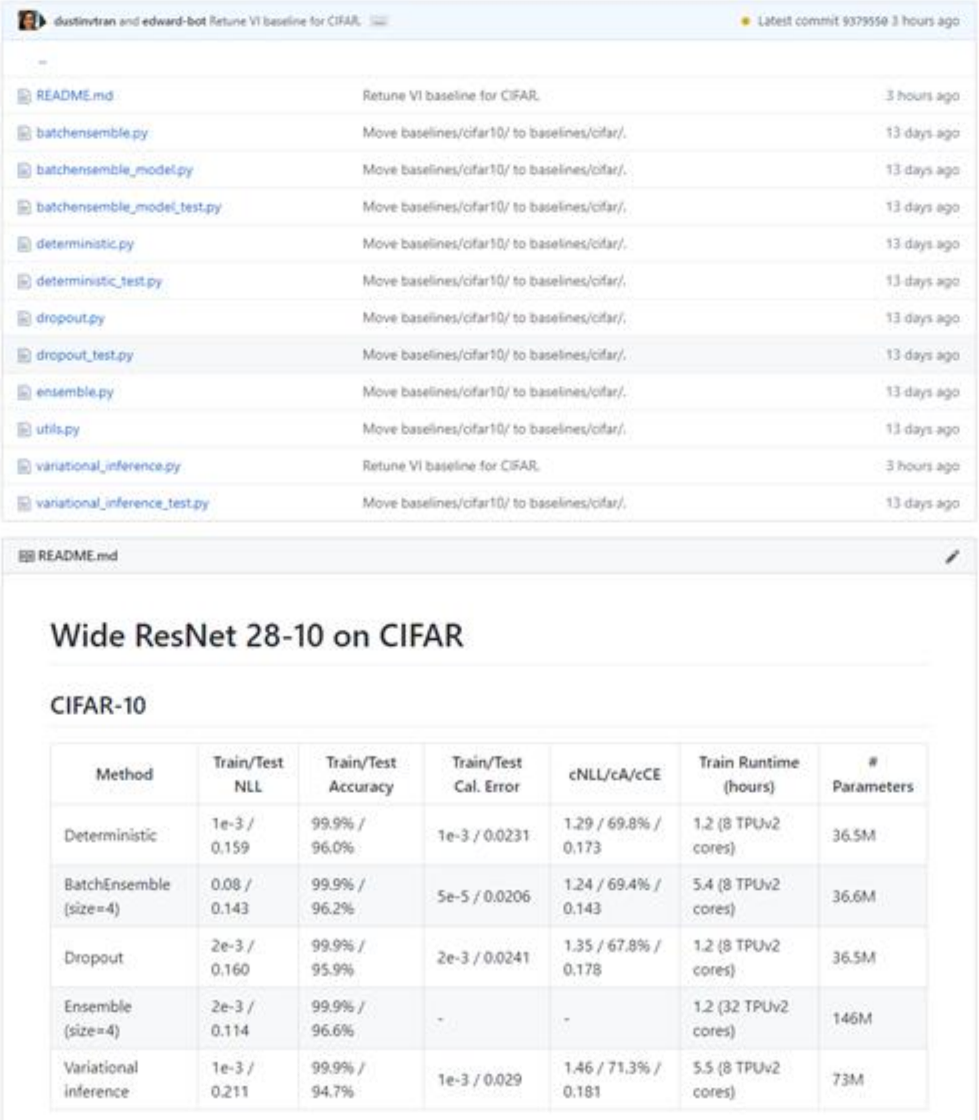
Ready for use: 7 settings, including:

- 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

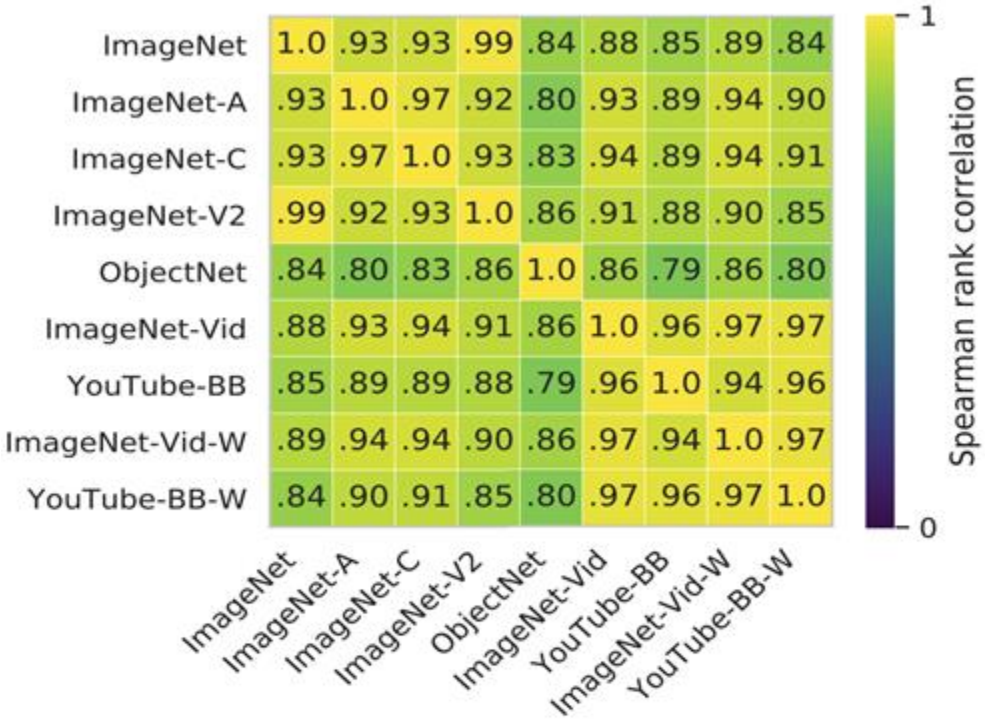
Robustness Metrics

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.

References

- Practical Uncertainty Estimation & Out-of-Distribution Robustness in Deep Learning
  - Video: https://slideslive.com/38935801/practical-uncertainty-estimation-outofdistribution-robustness-in-deep-learning
  - Link: https://neurips.cc/Conferences/2020/Schedule?showEvent=16649