Uncertainty estimation for neural network models

Alexey Zaytsev

Thanks for slides to

M. Panov K. Fedyanin, A. Fishkov and N. Kotelevskii

Sber

31.05.2021

Skoltech

Outline

1. Uncertainty Estimation in Machine Learning

2. Probabilistic Models and Uncertainty Types

3. Out of Distribution Detection for Neural Networks
    - Uncertainty Estimation by Disagreement
    - Bayesian Models
    - Improving Uncertainty Estimates for Single Deterministic Model

Outline

1. Uncertainty Estimation in Machine Learning

2. Probabilistic Models and Uncertainty Types

3. Out of Distribution Detection for Neural Networks
   - Uncertainty Estimation by Disagreement
   - Bayesian Models
   - Improving Uncertainty Estimates for Single Deterministic Model

Uncertainty Estimation: why should we care?

Goal: Provide the measure of uncertainty $\hat{\sigma}(\mathbf{x})$ of ML model prediction $\hat{f}(\mathbf{x})$ at a given point.

Use cases:

- Possibility of rejection to predict
- Out of distribution data detection
- Adversarial examples detection
- Active learning
- Bayesian optimization

Regression and Uncertainty Estimation

Some machine learning models along with

- approximation

$$\hat{f}(\mathbf{x}) \simeq f(\mathbf{x})$$

  can provide

- uncertainty estimation

$$\hat{\sigma}^2(\mathbf{x}) \simeq \mathbb{E}\big(\hat{f}(\mathbf{x}) - f(\mathbf{x})\big)^2.$$

Regression and Uncertainty Estimation

Some machine learning models along with

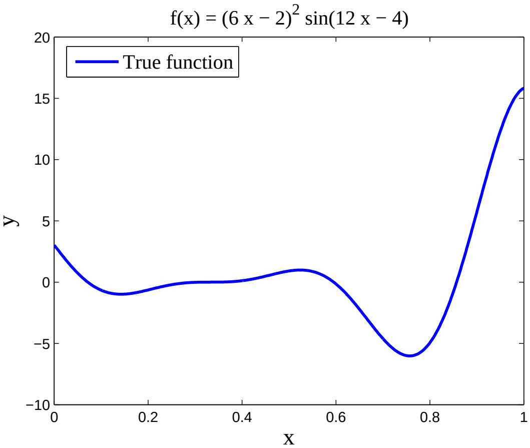
- approximation

$$\hat{f}(\mathbf{x}) \simeq f(\mathbf{x})$$

can provide

- uncertainty estimation

$$\hat{\sigma}^2(\mathbf{x}) \simeq \mathbb{E}\left(\hat{f}(\mathbf{x}) - f(\mathbf{x})\right)^2.$$



Regression and Uncertainty Estimation

Some machine learning models along with

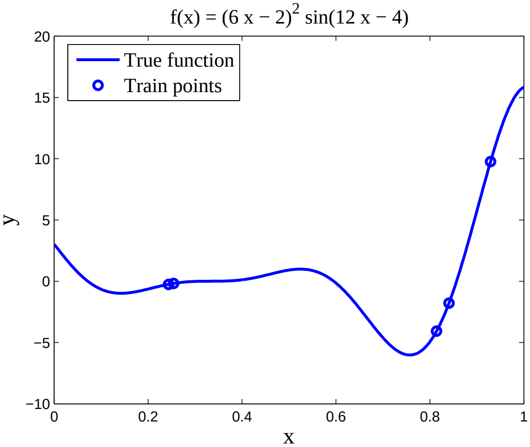
- approximation

$$\hat{f}(\mathbf{x}) \simeq f(\mathbf{x})$$

  can provide

- uncertainty estimation

$$\hat{\sigma}^2(\mathbf{x}) \simeq \mathbb{E}\left(\hat{f}(\mathbf{x}) - f(\mathbf{x})\right)^2.$$



Regression and Uncertainty Estimation

Some machine learning models along with

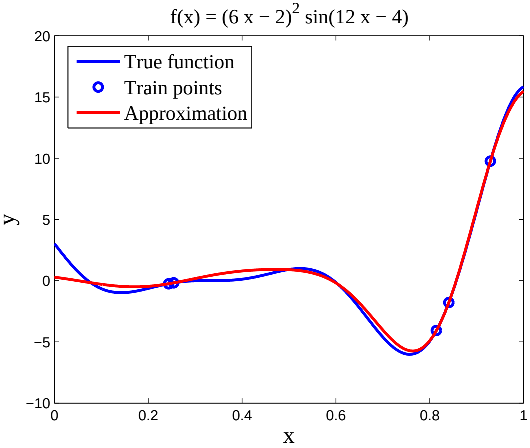
- approximation

$$\hat{f}(\mathbf{x}) \simeq f(\mathbf{x})$$

can provide

- uncertainty estimation

$$\hat{\sigma}^2(\mathbf{x}) \simeq \mathbb{E}\big(\hat{f}(\mathbf{x}) - f(\mathbf{x})\big)^2.$$



Regression and Uncertainty Estimation

Some machine learning models along with

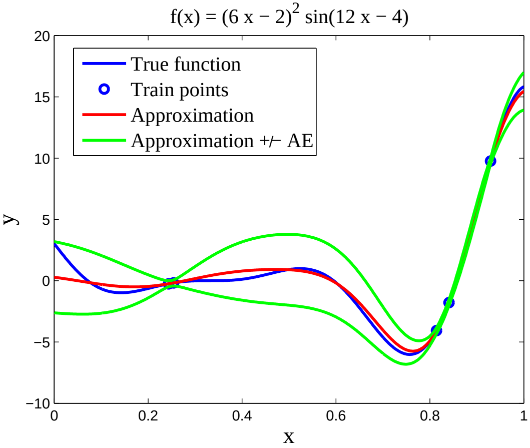
- approximation

$$\hat{f}(\mathbf{x}) \simeq f(\mathbf{x})$$

  can provide

- uncertainty estimation

$$\hat{\sigma}^2(\mathbf{x}) \simeq \mathbb{E}\left(\hat{f}(\mathbf{x}) - f(\mathbf{x})\right)^2.$$



Predictive Uncertainty Estimation

- The data: $D = \{X_i, Y_i\}_{i=1}^n$ – i.i.d from some distribution $P(x, y)$.
- Usually: $Y_i = f(X_i) + \varepsilon_i,\ i = 1, \dots, n$.
- The model $\hat{f}(\mathbf{x}) = \hat{f}(\mathbf{x} \mid D)$ is constructed based on the data D.
- Predictive confidence interval for confidence level $\alpha > 0$:

$$\mathbb{P}\Big(f(\mathbf{x}) \in \big[\hat{f}(\mathbf{x}) - c_\alpha(\mathbf{x}), \hat{f}(\mathbf{x}) + c_\alpha(\mathbf{x})\big]\Big) \geq 1 - \alpha.$$

Predictive Confidence Intervals for Linear Regression

- $Y_i = f(X_i) + \varepsilon_i, \; i = 1, \ldots, n.$
- $f(x) = \beta_0 + \beta_1 x; \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2).$
- Standard least squares estimates: $\hat{\beta}_0$ and $\hat{\beta}_1$.
- Under the model assumptions:

$$\hat{f}(x_*) = \hat{\beta}_0 + \hat{\beta}_1 x_* \sim \mathcal{N}\left(\beta_0 + \beta_1 x_*, \hat{\sigma}^2(x_*)\right),$$

  where $\hat{\sigma}^2(x_*) = \frac{\sigma^2}{n} \frac{\sum_{i=1}^n (X_i - x_*)^2}{\sum_{i=1}^n (X_i - \overline{X})^2}$.

- Thus, the confidence interval for confidence level $\alpha > 0$ is:

$$[\hat{\beta}_0 + \hat{\beta}_1 x_* \pm z_{\alpha/2} \cdot \hat{\sigma}(x_*)].$$

Machine Learning Models and Uncertainty Estimation

- General approaches:
  - Analytic statistical approaches (variance estimates and confidence intervals based on CLT);
  - Bootstrap.
- Bayesian inference
- Model-specific approaches:
  - Gaussian processes for regression and classification;
  - Neural networks with variance-predicting subnetwork;
  - Decision trees variance estimation at leaves.

Uncertainty in Classification

Classification models usually predict some score, that could be treated as confidence.
For example, in binary task confidence of positive class for logistic regression is just value of predicted probability:

$$p = f(x) = p\left(y = 1 \mid x\right).$$

It generalizes on multiclass task by using the value of class with maximum probability

$$p = \max_c p\left(y = c \mid x\right).$$

To get the uncertainty we should get the reverse value

$$f_{ue} = 1 - \max_c p\left(y = c \mid x\right).$$

Uncertainty in Classification

If we have many classes, the information entropy (Shannon, 1948) could be more expressive

$$\mathbb{H}(y \mid \mathbf{x}) = -\sum_{c} p(y = c \mid \mathbf{x}) \log p(y = c \mid \mathbf{x})$$

The problem with both maximum probability and entropy is that models happen to be overconfident

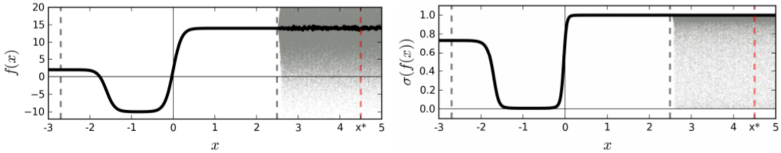


Image source: "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning" Yarin Gal et al, 2016

Calibration

We could estimate the overconfidence by expected calibration error on a validation set. The metric measures the difference in expectation between confidence and accuracy:

$$ECE = \mathbb{E}_{\hat{p}}[|\mathbb{P}(\hat{Y} = Y \mid \hat{p} = p) - p|].$$

- There are methods to improve model calibration, i.e. temperature scaling (TS).
- TS tweaks the single parameter of softmax temperature T.
- Given the logits values z, the new predictions will be

$$\hat{q} = \max_k \sigma_{\mathrm{SM}}\left(\mathbf{z}/T\right)^{(k)}.$$

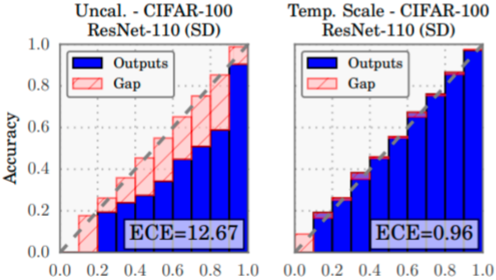


Image source: "On Calibration of Modern Neural Networks" Guo et al, 2017

Outline

1. Uncertainty Estimation in Machine Learning

2. Probabilistic Models and Uncertainty Types

3. Out of Distribution Detection for Neural Networks
   - Uncertainty Estimation by Disagreement
   - Bayesian Models
   - Improving Uncertainty Estimates for Single Deterministic Model

Probabilistic Models and Uncertainty Types

Probabilistic data generation:

$$(x_i, y_i) \sim p(x, y).$$

Joint probability density can be representeded in various ways:

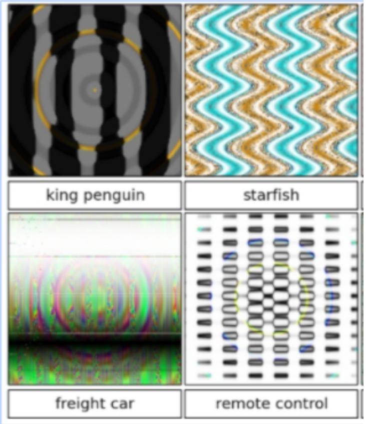
$$p(x, y) = p(y \mid x)p(x) = p(x \mid y)p(y),$$

where

- $p(y \mid x)$ – likelihood;
- $p(x)$ – covariate distribution;
- $p(x \mid y)$ – label-conditional covariate distribution;
- $p(y)$ – distribution of labels.

Train vs Test Data

- In distribution: $p_{\mathtt{test}}(x, y) = p_{\mathtt{train}}(x, y)$.
- Out-of-distribution: $p_{\mathtt{test}}(x, y) \neq p_{\mathtt{train}}(x, y)$.



Variants:

- **Covariate shift:** $p(x)$ changes, $p(y \mid x)$ is fixed.
- **Label shift:** $p(y)$ changes, $p(x \mid y)$ is fixed.
- **Open set recognition:** new classes are coming.

In this presentation we will stick to the term "OOD" for supervised problems as opposed to "anomaly" used for unsupervised problems (this is not standard terminology).

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

Epistemic vs Aleatoric Uncertainty

The prediction uncertainty can be decomposed into two terms: **aleatoric** and **epistemic**.

- **Aleatoric** uncertainty reflects noise in data.
  - It could be due to noisy labels, class overlap, or data ambiguity.
  - **This part of the predictive uncertainty <u>can not</u> be reduced when more data is given.**
- **Epistemic** uncertainty reflects lack of knowledge.
  - It is due to the total absence or just a few samples from a particular region.
  - **This part of the predictive uncertainty <u>can</u> be reduced when more data is given.**

Epistemic vs Aleatoric Uncertainty. Example

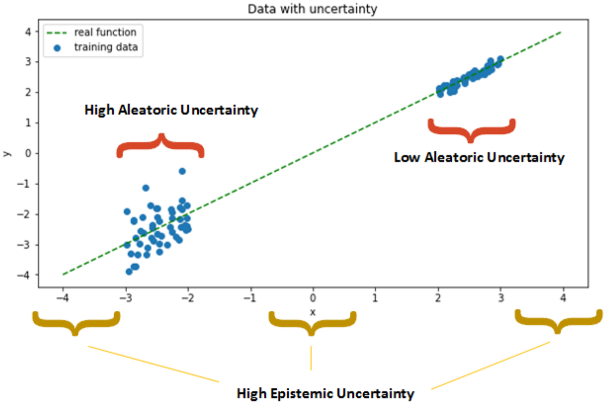


Figure: Regions, where different types of uncertainty are prevalent.

Image source: https://towardsdatascience.com/my-deep-learning-model-says-sorry-i-dont-know-the-answer-that-s-absolutely-ok-50ffa562cb0b

Epistemic vs Aleatoric Uncertainty. Example

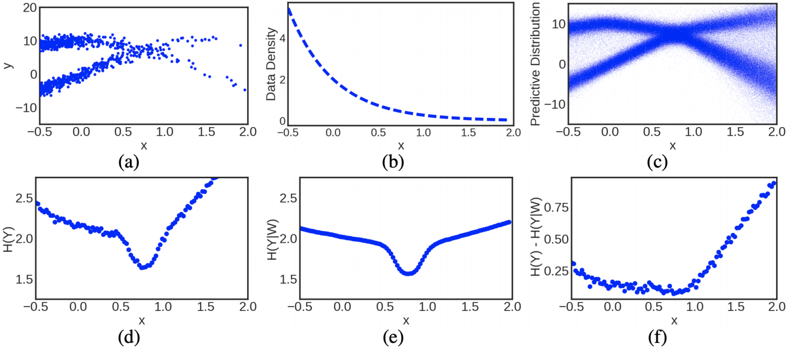


Figure: (a) True data; (b) Data density (c) Predictive distribution (d) Total uncertainty (e) Aleatoric uncertainty (f) Epistemic uncertainty

Image source: [Depeweg et al., 2018] Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning, ICML 2018.

Epistemic vs Aleatoric Uncertainty. Decomposition

It turns out, one could decompose the total predictive uncertainty mathematically. If we use **entropy** as an uncertainty measure, for a new data point x^* with predicted y^* it could be written as follows:

$$\underbrace{\mathbb{H}\ \mathbb{E}_{p(\theta|D)} p(y^* \mid x^*, \theta)}_{Total\ \ uncertainty} = \underbrace{\mathbb{E}_{p(\theta|D)} \mathbb{H} p(y^* \mid x^*, \theta)}_{Aleatoric\ \ uncertainty} + \underbrace{\mathbf{MI}(y^*, \theta \mid x^*, D)}_{Epistemic\ \ uncertainty}, \tag{1}$$

where

- D is a training dataset;
- θ denotes parameters of our models
  - different models in ensemble;
  - different weights, sampled from variational approximation
  - ...

Epistemic vs Aleatoric Uncertainty. Decomposition

It turns out, one could decompose the total predictive uncertainty mathematically. If we use **variance** as an uncertainty measure, for a new data point x^* with predicted y^* it could be written as follows:

$$\underbrace{\sigma^2(y^* \mid x^*)}_{Total \ \ uncertainty} = \underbrace{\mathbb{E}_{p(y^*|\theta,x^*)}\sigma^2(y^* \mid \theta, x^*)}_{Aleatoric \ \ uncertainty} + \underbrace{Var_{p(\theta|D)}\mathbb{E}_{p(y^*|\theta,x^*)}y^*}_{Epistemic \ \ uncertainty}. \quad (2)$$

where

- D is a training dataset;
- θ denotes parameters of our models
  - different models in ensemble;
  - different weights, sampled from variational approximation
  - ...

Epistemic vs Aleatoric Uncertainty. Conclusion

It is essential to separate two types of uncertainty, because

- The total **uncertainty could be high for both** regions, where epistemic or aleatoric uncertainty is high.

- **For OOD detection we are not interested in aleatoric** – it is "known-unknown". Thus, we have to have access to epistemic uncertainty ("unknown-unknown") to find OOD samples effectively.

- To compute epistemic uncertainty, we can use mutual information (in case of regression) or variance of the mean of predictive distribution (in case of classification).

Outline

1. Uncertainty Estimation in Machine Learning

2. Probabilistic Models and Uncertainty Types

3. Out of Distribution Detection for Neural Networks
   - Uncertainty Estimation by Disagreement
   - Bayesian Models
   - Improving Uncertainty Estimates for Single Deterministic Model

Uncertainty Estimation by Disagreement

- Some models provide multiple predictions for a single point (i.e. members of the ensemble or multiple passes for bayesian/dropout).
- In this case, we could use disagreement between models as a measure of uncertainty.
- One metric is mutual information, which is the difference between the entropy of the mean and means of the entropy (i.e., epistemic uncertainty):

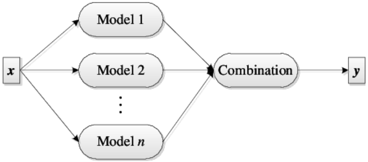
$$MI = \mathbb{E}_{p(\omega|)}\left[\sum_c p(y = c \mid \mathbf{x}, \omega) \log p(y = c \mid \mathbf{x}, \omega)\right] - \sum_c p\left(y = c \mid \mathbf{x}\right) \log p\left(y = c \mid \mathbf{x}\right).$$

- Another option is the variation ratio. It is defined as the proportion of cases with not most popular prediction:

$$\mathbf{v} := 1 - \frac{f_m}{N}.$$

Ensembles

- An ensemble of models is a group of models solving the same task, i.e. with same architecture, but different initialization.
- Ensemble prediction averaging is well-known way to increase the performance in ML.
- [Lakshminarayanan et al., 2017] showed that it could be used not only for accuracy but for the uncertainty estimation as well, both for classification and regression tasks.
- Further studies confirmed that ensembles outperform other methods in many cases.



x

Model 1

Model 2

⋮

Model n

Combination

y

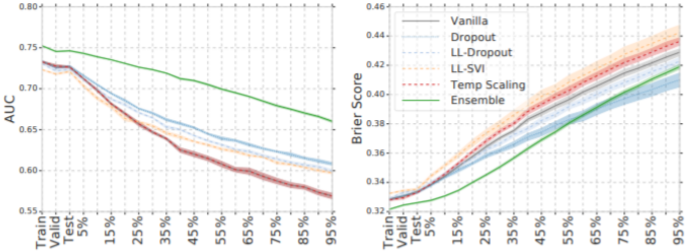


Image source: “Can you trust your model uncertainty” Ovadiv et al, 2019

Faster ensembling

- The main downside of ensembles is k-time overhead both in computation time and memory.
- The are attempts to decrease the computational cost and memory consumption
  - Monte-Carlo dropout [Gal and Ghahramani, 2016] allows to use a single model, but still require k-time computation overhead
  - DPP-based dropouts [Tsymbalov et al., 2020, Shelmanov et al.,] allow to diversify dropout and use only last layer dropout, significantly increasing the speed
  - Ensemble distribution distillation [Malinin et al., 2019] approximates the ensemble distribution with Dirichlet distribution
  - Some methods use different stage of training as ensemble members to speed up the training (snapshot ensembles [Huang et al., 2017], fast geometric ensembling [Garipov et al., 2018]).
- It is worth mentioning, that all methods beat the ensemble in computation cost, but they are inferior in performance.

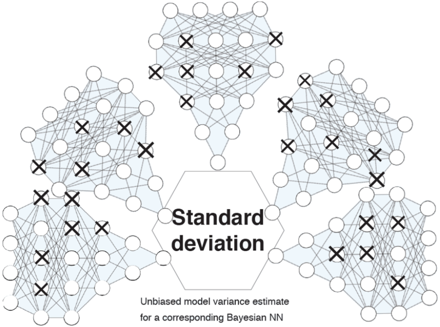
Uncertainty Estimation for Neural Networks

Problem: While simple for linear regression it might be hard to construct confidence intervals for more complex models.

Types of uncertainty estimates for Neural Networks:

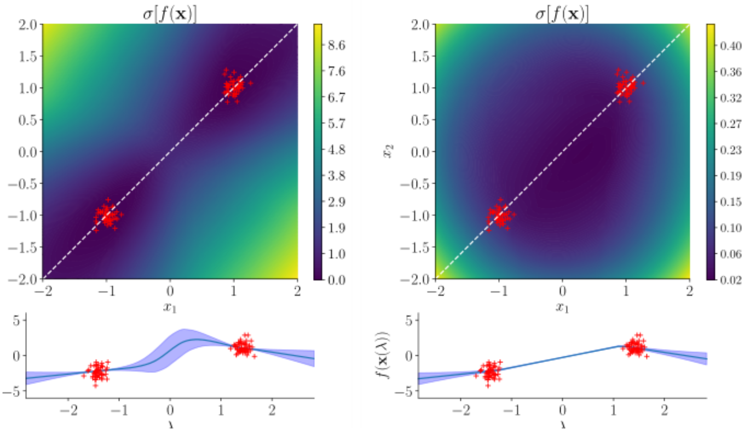
- Analytic estimates;
- Ensembling (NNs trained from different initializations);
- Bayesian Neural Networks;
- Dropout-based.

MC-Dropout
from [Gal and Ghahramani, 2016].


Standard deviation
Unbiased model variance estimate
for a corresponding Bayesian NN

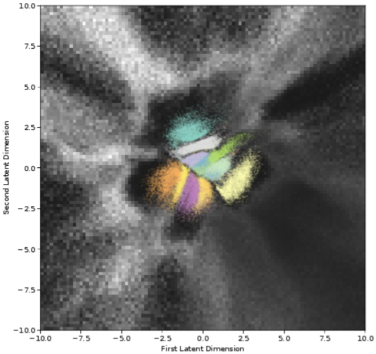
The problem with MC-Dropout

Overconfident predictions for out-of-sample points



The problem with MC-Dropout

Overconfident predictions for out-of-sample points



MNIST data in 2-dimensional latent space of autoencoder.

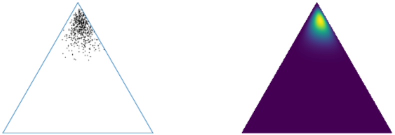
Prior Networks: introduction

- Consider a distribution $P(\mathbf{x}, \mathbf{y})$ over objects and labels and a finite training dataset $\mathcal{D}$
- From the Bayesian perspective, a trained model induces a posterior predictive distribution on the labels:

$$P(\mathbf{y}_0 \mid \mathbf{x}_0, \mathcal{D}) = \int P(\mathbf{y}_0 \mid \mathbf{x}_0, \theta) P(\theta \mid \mathcal{D}) d\theta. \tag{3}$$

  - For a classification task, the model returns a probability assignment for all classes and we get a distribution of such assignments – a distribution on the probability simplex.

- True posterior over the model parameters and the integral in (3) are both intractable for large models
  - It can be approximated using sampling (MC-dropout, ensembling).



(a) Ensemble (b) Distribution

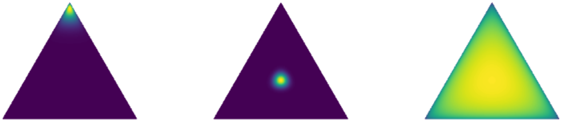
Prior Networks: goals

- Unlike previous approaches, here we want to explicitly parameterize a distribution on the probability simplex using a DNN (we focus on the classification task):

$$\begin{gathered}\boldsymbol{\mu} = [\mu_1, \ldots, \mu_K] = [P(y = c_1), \ldots, P(y = c_K)], \\ P(\boldsymbol{\mu} \mid \mathbf{x}_0, \mathcal{D}) = \int P(\boldsymbol{\mu} \mid \mathbf{x}_0, \theta) p(\theta \mid \mathcal{D}) d\boldsymbol{\mu}.\end{gathered} \tag{4}$$

- Still intractable, like (3). Use a point estimate:

$$P(\theta \mid \mathcal{D}) = \delta(\theta - \widehat{\theta}) \Rightarrow P(\boldsymbol{\mu} \mid \mathbf{x}_0, \mathcal{D}) \approx P(\boldsymbol{\mu} \mid \mathbf{x}_0, \widehat{\theta}) \tag{5}$$



(a) Confident Prediction (b) High data uncertainty (c) Out-of-distribution

Prior networks: model

- Use Dirichlet distribution as a distribution on class probabilities. A Dirichlet Prior Network will model parameters of this point-dependent distribution:

$$P(\mu \mid \mathbf{x}_0, \widehat{\theta}) = \mathtt{Dir}(\boldsymbol{\mu}, \alpha),\ \alpha = f(\mathbf{x}_0, \widehat{\theta}). \tag{6}$$

- Train the model:
  - Optimize the following functional:

$$\mathcal{L}(\theta) = \mathbb{E}_{P_{in}(\mathbf{x})} KL[\mathtt{Dir}(\boldsymbol{\mu} \mid \alpha_{in}) \mid P(\boldsymbol{\mu} \mid \mathbf{x}, \theta)] + \mathbb{E}_{P_{out}(\mathbf{x})} KL[\mathtt{Dir}(\boldsymbol{\mu} \mid \alpha_{out}) \mid P(\boldsymbol{\mu} \mid \mathbf{x}, \theta)]. \tag{7}$$

  - $\alpha_0 = \sum \alpha_{in,k}$ is a hyperparameter;
  - $\alpha_{in,k}$ may correspond to the point masses in the corners of the simplex;
  - $\alpha_{out} = \mathbf{1}$.

Deterministic model uncertainty capturing

By **deterministic** model, we imply a single model with a single set of weights.

- We will not be able to compute expectations w.r.t. model's parameters, thus have to use heuristics/proxies to separate aleatoric and epistemic uncertainties.
- The general idea of these methods - **use hidden feature representations learned by a model** to decide whether features extracted for a new unseen object are close to those from the training set or not.

In the following slides, several approaches which fall into this paradigm will be presented.

Deep Deterministic Uncertainty (DDU)

Why should it work?

- Convolution layers are known to be good feature extractors for images.
- They learn invariants over objects in the training dataset.
- Learned convolutions trigger a learned template on the input image, provide higher activations, or stay not-activated otherwise.
- If an object, different from the training dataset, is taken as an input, the final convolution's triggers will be different from typical training triggers.

Objects, different from training ones (OOD), should provide significantly different extracted features.

Deep Deterministic Uncertainty (DDU)

A model could map different objects into the same feature representations if there are no appropriate constraints. It is known as **feature collapse problem**.
To get rid of that, the model satisfy **bi-Lipschitz** constraint to be:

- **sensitive**, to distinguish between different objects
- **smooth**, to provide compact representations

$$K_1\|x_1 - x_0\| \leq \|f_\theta(x_1) - f_\theta(x_2)\| \leq K_2\|x_1 - x_0\| \tag{8}$$

It could be satisfied with appropriate **inductive biases**:

- Spectral normalization (to make model Lipschitz from above). Other options like gradient penalty or weight clipping are also suitable,
- Skip connections (ResNet architecture) to satisfy Lipschitz from below.

Deep Deterministic Uncertainty (DDU)

Next step (proposed in [Mukhoti et al., 2021]) – fit Gaussian Mixture Model (GMM) in the space of extracted features (another option – in the space of logits) with the number of components, equal to the number of classes.

The proxy to **epistemic uncertainty** would be the density under trained GMM:

$$\log p(x^*_{features}) = \log \sum_{i=1}^{N_{cl}} p(x^*_{features} | \mu_i, \sigma_i^2) p(c_i), \tag{9}$$

where μ_i, σ_i^2 are parameters of a Gaussian distributions, associated with c_i-th class. All classes are assumed to have the same probability, thus $p(c_i) = \frac{1}{N_{cl}}$.
As for **aleatoric** uncertainty – it is expressed as a predictive entropy/variance when an object is mapped to a high-density region.

On the validation set, we select a threshold to detect if the object is an outlier or not.

Deep Uncertainty Quantification(DUQ)

This method uses the same idea of extracting features but utilizes **non-parametric** model to estimate the density of an object in feature space.

Specifically, they use the RBF kernel to capture epistemic uncertainty:

$$K_c(f_\theta(x), e_c) = \exp\left[-\frac{\frac{1}{n}\|W_c f_\theta(x) - e_c\|^2}{2\sigma^2}\right], \tag{10}$$

where e_c – a centroid of a corresponding class and W_c – a weight matrix, learned for each class.

The loss function is used for the method is given by:

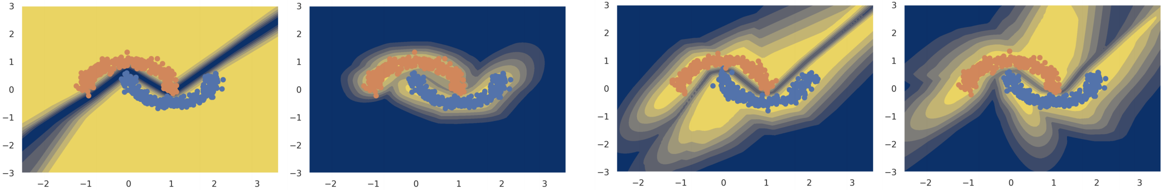
$$\mathrm{L}(x, y) = -\sum_c y_c \log(K_c) + (1 - y_c) \log(1 - K_c) \tag{11}$$

The class centroids are updated using an exponential moving average of the feature vectors of data points belonging to that class.

Deep Uncertainty Quantification(DUQ)

As in DDU, in DUQ we have to regularize a network to provide sensitive and smooth feature encoding. Authors are using ResNet architecture **with gradient penalty** to satisfy the bi-Lipschitz constraint.

The authors stress that a network must be *bi*-Lipschitz, illustrating it on the example: From left to right: **Deep ensembles, bi-Lipschitz network (DUQ), No Lipschitz regularization (DUQ), one-side Lipschitz regularization (DUQ)**



[Van Amersfoort et al., 2020] Uncertainty estimation using a single deep deterministic neural network. ICML, 2020.

Conclusions and Outlook

Summary:

- it is important to correctly model different sources of uncertainty;
- out-of-distribution data detection is a challenging task;
- its solution requires not only the development of the special algorithms
- but also the careful work with the model architecture and training procedure.

Areas to grow:

- improvement of uncertainty estimation methods based on single deterministic network;
- training of prior networks without OOD data.

Thank you for your attention!

References

Depeweg, S., Hernandez-Lobato, J.-M., Doshi-Velez, F., and Udluft, S. (2018).
Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning.
In *International Conference on Machine Learning*, pages 1184–1193. PMLR.

Gal, Y. and Ghahramani, Z. (2016).
Dropout as a bayesian approximation: Representing model uncertainty in deep learning.
In *Proc. ICML'16*, pages 1050–1059.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. (2018).
Loss surfaces, mode connectivity, and fast ensembling of dnns.
In *Advances in Neural Information Processing Systems*, pages 8789–8798.

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. (2017).
Snapshot ensembles: Train 1, get m for free.
arXiv preprint arXiv:1704.00109.

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

Malinin, A., Mlodozeniec, B., and Gales, M. (2019).
Ensemble distribution distillation.
arXiv preprint arXiv:1905.00076.

Mukhoti, J., Kirsch, A., van Amersfoort, J., Torr, P. H., and Gal, Y. (2021).