Data Science Lab @Theresis

Uncertainties for classification tasks in Deep Neural Networks: a last layer approach

Nicolas BROSSE
nicolas.brosse@thalesgroup.com

11 May 2020

- Joint work with Carlos Riquelme (Google Brain Zurich), Alice Martin (CMAP, Ecole Polytechnique), Sylvain Gelly (Google Brain Zurich) , Éric Moulines (CMAP, Ecole Polytechnique).
- Presentation based on `https://arxiv.org/abs/2001.08049`.

Contents

Motivation and approach

Introduction

Facts:

- Deep Neural Networks are accurate on average, and their predictions are usually right.
- At the individual data-point level, what is the *confidence* of the model in its own prediction ?

Consequences:

- Deep models are currently deployed in scenarios where making mistakes is cheap.
- For critical uses-cases, we need to develop systems that are able to say “I don’t know”.

Introduction

- This work is part of a vast movement in academic/industrial research looking for a more robust AI.
- Focus on classification tasks.
- Difference with interpretability methods: we are looking for a confidence score associated to a prediction that enables to quickly identify problematic inputs.
- Main goal is to automate decision making while providing strong risk guarantees.

Bayesian framework ?

- A principled approach to do probabilistic inference [HvC93, Nea96, BB98].
- But, at the scale of modern deep neural networks, Bayesian methods face serious computational issues [GG16, LPB17].
- A recent article, [WRV^{+}20], examines Bayesien posteriors in Deep Neural Networks.

A last layer approach

Basic idea:

1. train end-to-end a deep classifier on input-output pairs (x, y) to obtain an accurate task-dependent representation z of the data,
2. fit an ensemble of models on (z, y). The simplicity of this new dataset allows to compute explicit uncertainty estimates.

We explore four concrete instances of uncertainty algorithms based on

- Stochastic Gradient Descent (SGD) [MHB17],
- Stochastic Gradient Langevin Dynamics (SGLD) [WT11],
- the Bootstrap, see Section 8.4 of [FHT01],
- Monte Carlo Dropout [GG16].

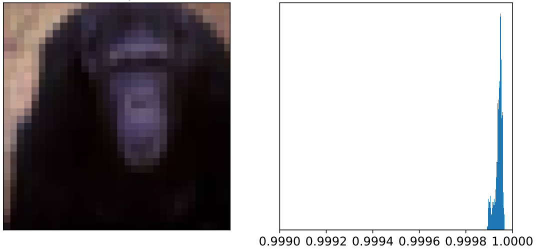
A last layer approach

- Core idea has some connections with transfer learning [YCBL14, RASC14, DJV$^+$14].
- By sequentially tackling two tasks (representation learning and uncertainty quantification), these algorithms performed on the last layer of the neural networks reduce the computational cost.
- **Main take-home message**: there is limited value in adding multiple uncertainty layers to high-level representations in deep classifiers.

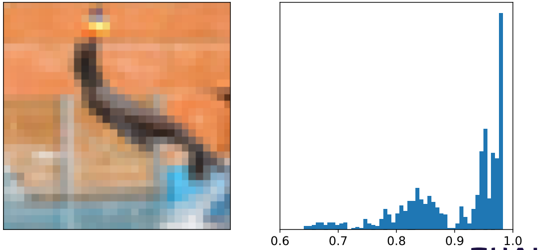
Be careful with softmax values

Histograms of $p_{\max}$ values given by Stochastic Gradient Langevin Dynamics (SGLD) on top of a pre-trained VGG-16 network on CIFAR-100, $\{\max_k p\,(k|x, \theta_i)\}_{i=1}^{n_{\text{samples}}}$.
Chimpanzee image, average $\bar{p}_{\max} = 0.9999$.
Seal image, wrongly classified as a worm. Class is predicted with a high average softmax ($\bar{p}_{\max} = 0.8985$).

Correct class: chimpanzee, 0.9999.



Wrong class: worm (true: seal), 0.8985.



Uncertainty metrics

Uncertainty metrics

Problem description

- $\mathcal{X}$ is a feature space, $\mathcal{Y} = \{1, \ldots, K\}$ a finite label set with $K \geq 2$ classes.
- Training dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N} \subset (\mathcal{X}, \mathcal{Y})^N$ of N points independently distributed according to a pair of random variables (X, Y).
  Test set: $\mathcal{T} = \{x_i, y_i\}_{i=1}^{n_{\text{test}}}$.
- Network parameters (weights and bias) are denoted by θ.
- Network trained using variants of stochastic gradient descent with the cross-entropy loss.
- $\{p\left(k|x_i, \theta\right)\}_{k=1}^{K}$ is the output probability distribution over $\mathcal{Y}$ predicted by the network.
- The classifier $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ is generally obtained by taking the argmax, $f_\theta(x) = \arg\max_{k \in \{1,\ldots,K\}} p\left(k|x, \theta\right)$ for $x \in \mathcal{X}$.

Uncertainty metrics

Uncertainty metrics

Calibration

- A model is calibrated if, on average over input points $x \in \mathcal{X}$, the predicted distribution $\{p(k|x,\theta)\}_{k=1}^{K}$ does match the true underlying distribution over the K classes.
- In most works, the authors focus on $p_{\max}(x,\theta) = \max_{k \in \{1,\ldots,K\}} p(k|x,\theta)$ matching only.
- Modern neural networks are often miscalibrated. Simple methods exist to alleviate this issue, such as temperature scaling, [GPSW17].
- Calibrated neural networks are important for model interpretability.
- They do not offer a systematic and automated way to neither improve accuracy nor detect out-of-distribution samples.

Selective classification

- Selective classification, also known as abstention, is not restricted to deep learning, [BW08, CDM16, GECd18].
- A selective classifier is a pair (f, r) where f is a classifier, and $r : \mathcal{X} \rightarrow \{0, 1\}$ is a selection function which serves as a binary qualifier for f.
- The selective classifier abstains from prediction at a point $x \in \mathcal{X}$ if $r(x) = 0$, and outputs $f(x)$ when $r(x) = 1$.

Selective classification

- The performance of a selective classifier can be quantified using the notions of coverage and selective risk.
- The coverage is defined as $\mathrm{cov}(r) = \mathbb{E}\left[r(X)\right]$, whereas selective risk is given by

$$\mathrm{srisk}(f, r) = \frac{\mathbb{E}\left[\mathbb{1}\left\{Y \neq f(X)\right\} r(X)\right]}{\mathbb{E}\left[r(X)\right]} \, .$$

- Their empirical estimations over the test set $\mathcal{T}$ are:

$$\mathrm{cov}_{n_{\mathrm{test}}}(r) = \frac{1}{n_{\mathrm{test}}} \sum_{i=1}^{n_{\mathrm{test}}} r(x_i) \, ,$$

$$\mathrm{srisk}_{n_{\mathrm{test}}}(f, r) = \frac{\sum_{i=1}^{n_{\mathrm{test}}} \mathbb{1}\left\{y_i \neq f(x_i)\right\} r(x_i)}{\sum_{i=1}^{n_{\mathrm{test}}} r(x_i)} \, .$$

Selective classification

- A natural way to define a selection function r is by means of a *confidence* function $\kappa : \mathcal{X} \to \mathbb{R}$ which quantifies how much we trust the $f(x)$ prediction for input x.
- The selection function r is then constructed by thresholding κ, *i.e.* given $s \in \mathbb{R}$, for all $x \in \mathcal{X}$, we set $r_s(x) = \mathbb{1}\{\kappa(x) \geq s\}$. We only classify x if its confidence is at least s.

Selective classification

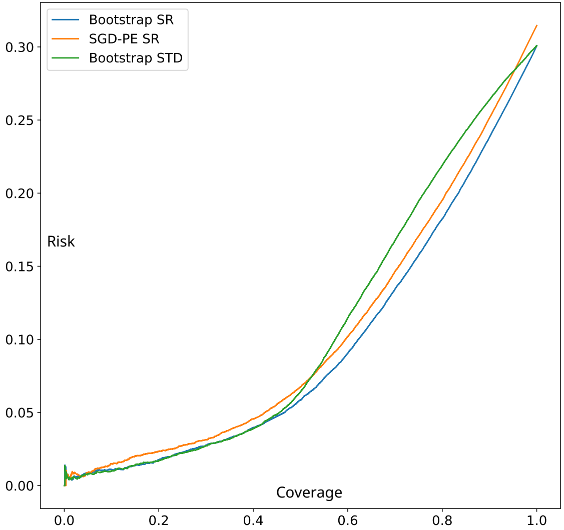
- Let $\mathcal{S}$ be the set of all κ values for those points in the test dataset $\mathcal{T}$, $\mathcal{S} = \{\kappa(x), x \in \mathcal{T}_x\}$.
- The performance of confidence function κ can be measured using the Area Under the Risk-Coverage curve (AURC) computed over $\mathcal{T}$:

$$\mathrm{AURC}(f, \kappa) = \frac{1}{n_{\mathrm{test}}} \sum_{s \in \mathcal{S}} \mathrm{srisk}_{n_{\mathrm{test}}}(f, r_s) \ .$$

- Better confidence functions lead to a faster decrease of the associated risk when we decrease coverage, which results in a lower AURC.

Selective classification

Risk-Coverage curve on ImageNet.



Out-of-distribution detection

- Out-of-distribution detection = find out when a data point is not drawn from the training data distribution.
- See the preprint for the results on this metric.

Uncertainty metrics

Confidence functions

Quick detour to Bayesian statistics

- In the Bayesian framework, a major obstacle often encountered in practice is to sample from the *posterior distribution* $\theta \mapsto p\,(\theta|\mathcal{D})$.
- Approximation using workarounds such as variational inference [WJ08], or Markov Chain Monte Carlo algorithms, see e.g. Chapter 11 of [GSC$^+$13].
- The predictive posterior distribution is defined by

$$p\,(y|x) = \int_{\Theta} p\,(y|x,\theta)\, p\,(\theta|\mathcal{D})\, \mathrm{d}\theta \ .$$

- In practise:

$$\hat{p}\,(y|x) = \frac{1}{n_{\text{samples}}} \sum_{i=1}^{n_{\text{samples}}} p\,(y|x,\theta_i) \ ,$$

where $\{\theta_i\}_{i=1}^{n_{\text{samples}}}$ are approximately drawn according to the posterior distribution.

Softmax Response (SR)

- $\mathrm{SR}(x) = \max_{k\in\{1,\ldots,K\}} p\left(k|x\right)$ where $\{p\left(k|x\right)\}_{k=1}^{K}$ is the predictive posterior distribution.
- Estimation:
$$\widehat{\mathrm{SR}}(x) = \max_{k\in\{1,\ldots,K\}} \hat{p}\left(k|x\right) \ .$$
- Associated classifier: $f(x) = \arg\max_{k\in\{1,\ldots,K\}} p\left(k|x\right)$ and empirical estimation $\hat{f}(x) = \arg\max_{k\in\{1,\ldots,K\}} \hat{p}\left(k|x\right)$.

Standard deviation of the posterior distribution

- Classifier fixed: $f(x) = \arg\max_k p\,(k|x)$.
- Standard deviation of the probability at $f(x)$ under the posterior:

$$\mathrm{STD}^2(x) = \int_\Theta p\,(f(x)|x,\theta)^2\, p\,(\theta|\mathcal{D})\,\mathrm{d}\theta - \left(\int_\Theta p\,(f(x)|x,\theta)\, p\,(\theta|\mathcal{D})\,\mathrm{d}\theta\right)^2 .$$

- Estimation:

$$\widehat{\mathrm{STD}}^2(x) = \frac{1}{n_{\text{samples}}}\sum_{i=1}^{n_{\text{samples}}} p\left(\hat{f}(x)\middle|x,\theta_i\right)^2 - \left(\frac{1}{n_{\text{samples}}}\sum_{i=1}^{n_{\text{samples}}} p\left(\hat{f}(x)\middle|x,\theta_i\right)\right)^2 ,$$

$\{\theta_i\}_{i=1}^{n_{\text{samples}}}$ are approximately drawn according to the posterior distribution.
- Confidence function is defined as $\kappa(x) = -\,\mathrm{STD}(x)$.

Entropy of q

- Probability distribution over the K classes defined as

$$q\left(k|x\right) = \int_{\Theta} \mathbb{1}\left\{f_\theta(x) = k\right\} p\left(\theta|\mathcal{D}\right) \mathrm{d}\theta \ ,$$

  where $f_\theta(x) = \arg\max_{k\in\{1,\ldots,K\}} p\left(k|x,\theta\right)$

- Idea of q: measure the amount of posterior mass under which each class is selected.
- Estimation:

$$\hat{q}\left(k|x\right) = \frac{1}{n_{\text{samples}}} \sum_{i=1}^{n_{\text{samples}}} \mathbb{1}\left\{f_{\theta_i}(x) = k\right\} \ .$$

- Confidence is based on the *entropy* of $\left\{q\left(k|x\right)\right\}_{k=1}^{K}$: $\kappa(x) = -\mathcal{H}(q\left(\cdot|x\right))$.

Algorithms

Last layer approach

- Goal: approximately draw samples from the posterior distribution $\theta \mapsto p(\theta|\mathcal{D})$.
- Core idea: explicitly disentangling representation learning and uncertainty estimation.
- $\mathcal{D}$: classification training dataset. We train a standard deep neural network to convergence using the cross entropy loss and a classical optimizer.
- θ^* = the parameters of the trained network after convergence.
- $\mathcal{Z}$ = vector space containing the input to the last layer of the trained neural network.
- Compute the last layer features $z \in \mathcal{Z}$ from the inputs $x \in \mathcal{X}$ by making a forward pass through the trained network.

Last layer approach

- We produce a new training dataset $\mathcal{R} = \{z_i, y_i\}_{i=1}^{N}$ which should provide a simpler representation of the data for the classification task.
- Last layer of the network is a dense layer θ with a softmax activation, *i.e.* for $\theta = (W, b)$

$$\{p\,(k|z,\theta)\}_{k=1}^{K} = \mathrm{softmax}\,(W\,z + b)\ \ .$$

- Uncertainty estimation is carried out on $\mathcal{R}$ via any algorithm that computes confidence estimates. They all compute an *ensemble* of models $\{\theta_i\}_{i=1}^{n_{\mathrm{samples}}}$.

Stochastic Gradient Langevin Dynamics (SGLD)

- Stochastic Gradient Langevin Dynamics (SGLD) is a Monte Carlo Markov Chain (MCMC) algorithm [WT11], adapted from the Langevin algorithm [RT96] to large-scale datasets.
- by the Bayes' rule, posterior distribution:

$$\theta \mapsto p\left(\theta|\mathcal{D}\right) \propto p\left(\theta\right) \prod_{i=1}^{N} p\left(y_i|z_i, \theta\right)$$

where $\theta \mapsto p\left(\theta\right)$ is a prior distribution on θ (Gaussian prior in practice).

Stochastic Gradient Langevin Dynamics (SGLD)

Update equation of SGLD:

$$\theta_{k+1} = \theta_k + \gamma \left(\frac{1}{s} \sum_{i \in \mathcal{S}} \nabla \log p\left(y_i | z_i, \theta_k\right) + \frac{\nabla \log p\left(\theta_k\right)}{N} \right) + \sqrt{\frac{2\gamma}{N}} Z_{k+1} \ ,$$

where

- γ is a constant learning rate,
- $\mathcal{S}$ a mini batch from $\mathcal{R}$ of size $s \in \mathbb{N}^*$,
- $(Z_k)_{k \in \mathbb{N}^*}$ an i.i.d. sequence of standard Gaussian random variables.

THALES

Stochastic Gradient Langevin Dynamics (SGLD)

- SGLD applied with a constant learning rate γ.
- Thinning technique to reduce the memory cost: given a thinning interval $n_{\text{thinning}} \in \mathbb{N}^*$ and a number of samples $n_{\text{samples}} \in \mathbb{N}^*$, we run the Markov chain $(\theta_k)_{k\in\mathbb{N}}$ during $n_{\text{samples}} \times n_{\text{thinning}}$ steps and at every n_{thinning} iteration, we save the current parameters of the (last layer or full) neural network θ.

Stochastic Gradient Descent (SGD)

- update equation of SGLD = update equation of Stochastic Gradient Descent (SGD), apart from the addition of the Gaussian noise $\sqrt{2\gamma/N}Z$.
- [MHB17] shows that, under certain assumptions, SGD with a carefully chosen constant step-size can be seen as approximate sampling from a posterior distribution with an appropriate prior.

SGLD and SGD

Algorithm 1 SGLD and SGD

```
Input: data $\mathcal{R}$, neural network $\theta$, number of samples $n_{\text{samples}}$, thinning interval $n_{\text{thinning}}$, batch size $s$, learning rate $\gamma$.
Initialize $\theta = \theta^*$.
for $i = 1$ to $n_{\text{samples}}$ do
    for $j = 1$ to $n_{\text{thinning}}$ do
        $\theta \leftarrow \text{SGLD}(\theta, \gamma, s)$ or $\text{SGD}(\theta, \gamma, s)$
    end for
    Save $\theta$.
end for
```

Monte-Carlo Dropout

- Monte Carlo Dropout approximately samples from the posterior distribution $\theta \mapsto p(\theta|\mathcal{D})$ when applied at inference time [GG16].
- Widely used in practical applications [ZL17, LAA$^+$17, NPAA18].
- Dropout randomly sets a fraction $p_{\text{drop}} \in (0, 1)$ of input units to 0 at each update during training time, or at each forward pass during test time.
- We interleave a dropout layer after each max pooling layer in the VGG-type neural network and before each dense layer.

Monte-Carlo Dropout

Algorithm 2 MC-Dropout

```
**Input:** data $\mathcal{R}$, neural network $\theta$, number of samples $n_{\text{samples}}$, dropout rate $p_{\text{drop}}$, batch size $s$, learning rate $\gamma$, number of training epochs $n_{\text{epochs}}$.
**Initialize** $\theta = \theta^*$.
**Train** $\theta$, using SGD with a learning rate $\gamma$, batch size $s$, dropout rate $p_{\text{drop}}$ and a number of epochs $n_{\text{epochs}}$.
**Save** $\theta$.
For a given input $x$, we run $n_{\text{samples}}$ forward passes from $\theta$ using dropout again.
```

Bootstrap

- Crossroad between the Bayesian and the frequentist approaches [Efr12a, Efr12b].
- Sample with replacement N data points from the training dataset $\mathcal{R}$, and generate a new bootstrapped dataset $\mathcal{R}_B$.
- The last layer (multinomial logistic regression) or a full neural network is trained on $\mathcal{R}_B$ until convergence, and the parameters of the network θ are saved.
- Repeat this as many times as models needed, and then compute their ensemble.

Bootstrap

Algorithm 3 Bootstrap

```
Input: data $\mathcal{R}$, neural network $\theta$, number of samples $n_{\text{samples}}$, batch size $s$, learning rate $\gamma$, number of training epochs $n_{\text{epochs}}$.
for $i = 1$ to $n_{\text{samples}}$ do
    Initialize $\theta = \theta^*$.
    Sample a bootstrapped dataset $\mathcal{R}_B$ from $\mathcal{R}$.
    Train $\theta$ on $\mathcal{R}_B$, using SGD with a learning rate $\gamma$, batch size $s$ and a number of epochs $n_{\text{epochs}}$.
    Save $\theta$.
end for
```

THALES

Experimental results

Description

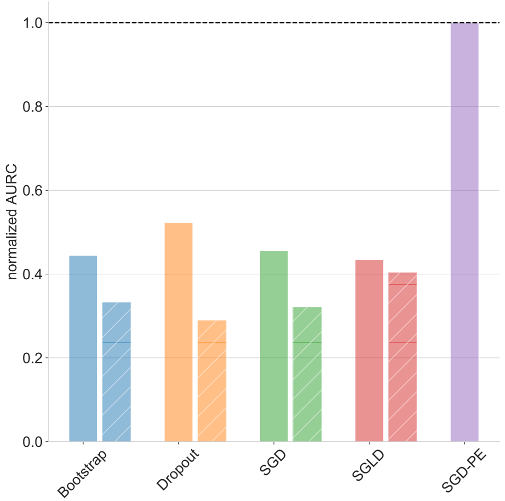
- Datasets: MNIST, CIFAR-10, CIFAR-100, and ImageNet.
- Classical models: fully connected for MNIST, VGG-16 for CIFAR, and NASNet for ImageNet.
- Four algorithms: MC-Dropout, Bootstrap, SGD and SGLD.
- Baseline algorithm: SGD-Point Estimate (SGD-PE) which computes the softmax outputs provided by the pretrained neural network. Only one confidence function for SGD-PE = the softmax response SR.

Description

- The algorithms are run both on the last layer and on the full neural networks for MNIST and CIFAR-10/100.
- Given the size of both ImageNet and the NASNet network, we assess the potential benefit of multiple uncertainty layers on ImageNet by adding up to 3 dense hidden layers with 4032 neurons on top of NASNet.
- min AURC = minimum value achieved using either SR, STD or the entropy of $\hat{q}$ as a confidence function.
- normalized AURC = ratio of min AURC over the AURC of SGD-PE (unique, using SR as confidence function)
- the lower is the AURC, the better is the result.

AURC and MNIST

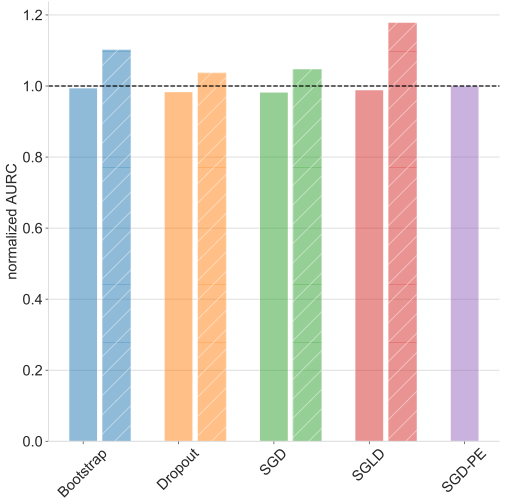
Normalized AURC for last-layer (solid) and full network (striped) versions of the four algorithms: Bootstrap, MC-Dropout, SGD, SGLD, and SGD-PE baseline, on MNIST.



THALES

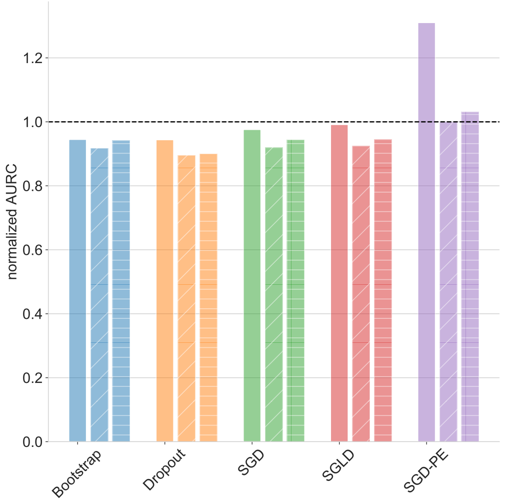
AURC and CIFAR-100

Normalized AURC for last-layer (solid) and full network (striped) versions of the four algorithms: Bootstrap, MC-Dropout, SGD, SGLD, and SGD-PE baseline, on CIFAR-100.



AURC and ImageNet

Normalized AURC for the 1 (solid), 2 (45-degree stripes) to 3 (horizontally striped) dense layer(s) versions of the algorithms: Bootstrap, MC-Dropout, SGD, SGLD and SGD-PE baseline, on ImageNet. The normalized AURC is based on the AURC obtained using SGD-PE on 2 dense layers on top of NASNet.



Summary of results

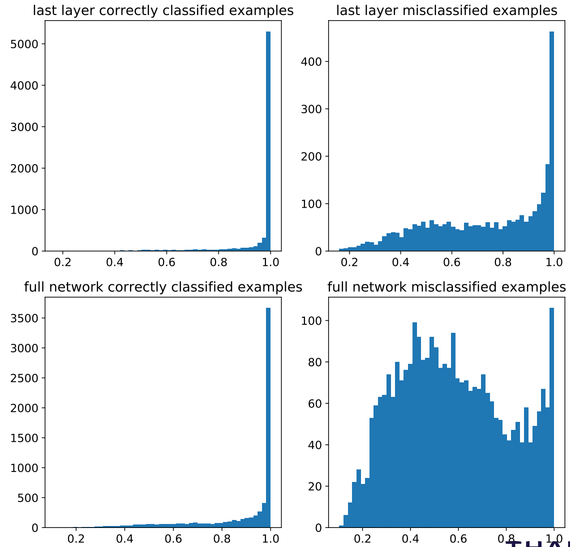
1. Adding multiple uncertainty layers does not help.
2. Softmax Response (SR) is a strong confidence function.
3. SGD Point-Estimate is actually a strong baseline.
4. SGLD is unstable on the full network.

Adding multiple uncertainty layers does not help

- Except on the MNIST dataset, where adding an extra hidden uncertainty layer improves the AURC, the last layer and its full network counterpart seem to perform similarly well for the four algorithms.
- In the case of MNIST, the histograms for correctly classified points are similar for both last-layer and full-network SGD versions. However, the full-network exhibits a greater dispersion for incorrectly classified points (see scale of y-axis). Both facts combined lead to a stronger AURC for the full-network algorithm, as it can better tell the difference between both sets of points.
- A different behavior can be observed on CIFAR-100, where the classification task is more difficult. The histograms of the full-network SGD are more dispersed for *both* correctly classified and misclassified points. In particular, as opposed to the MNIST scenario, a number of correctly classified points are no longer mapped to a high SR.

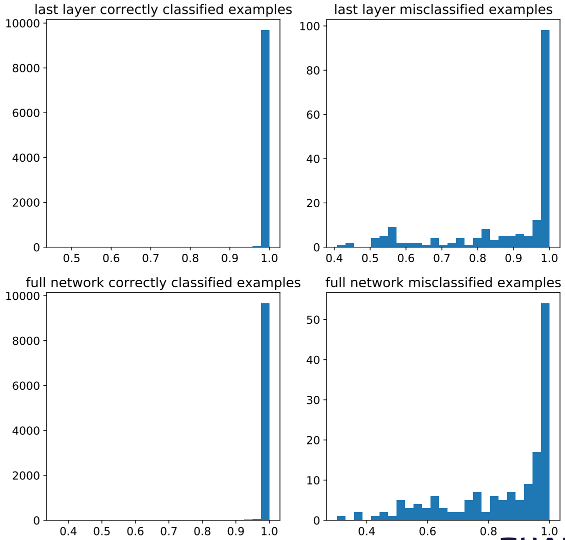
Histograms for CIFAR-100

Histograms of the $\widehat{\mathrm{SR}}$ confidence function values for all correctly classified and misclassified test data points. x-axis: $\widehat{\mathrm{SR}}$ values. y-axis: frequency per bin. **Top row:** Last layer version of SGD on CIFAR-100. **Bottom row:** Full network version of SGD on CIFAR-100. **Left column:** Correctly classified test data points. **Right column:** Misclassified test data points.



Histograms for MNIST

Similar plot, but for the MNIST dataset.



Softmax Response is a strong confidence function

- We compared several confidence functions: SR, STD and the entropy of $\hat{q}$.
- The softmax response SR does consistently outperform all the other confidence functions.

SGD Point-Estimate is a strong baseline

- SGD-PE is particularly competitive on CIFAR-10/100.
- Its main advantage is simplicity: it can be applied off-the-shelf and no two-stage procedure is needed.
- However, the method suffers in both MNIST and ImageNet, compared to the other algorithms. *Ensemble* techniques may bring additional stability and robustness in this context.

SGLD is unstable on the full network

- If the learning rate is not very small, SGLD tends to diverge, *i.e.* the accuracy (resp. the loss) decreases (resp. increases) over the iterations on the full network.
- Not visible when SGLD is only applied on the *last layer* of the neural network.
- In that scenario, the logarithm of the posterior distribution $\theta \mapsto p(\theta|\mathcal{D})$ is a *strongly log concave* function.

Thank you for your attention

Any questions ?

References I

[BB98] **D. Barber and Christopher Bishop.**
Ensemble learning in bayesian neural networks.
In *Generalization in Neural Networks and Machine Learning*, pages 215–237. Springer Verlag, January 1998.

[BW08] **Peter L. Bartlett and Marten H. Wegkamp.**
Classification with a reject option using a hinge loss.
J. Mach. Learn. Res., 9:1823–1840, June 2008.

[CDM16] **Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri.**
Boosting with abstention.
In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1660–1668. Curran Associates, Inc., 2016.

References II

[DJV+14] **Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell.**
Decaf: A deep convolutional activation feature for generic visual recognition.
In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 647–655, Bejing, China, 22–24 Jun 2014. PMLR.

[Efr12a] **Bradley Efron.**
A 250-year argument: Belief, behavior, and the bootstrap.
Bulletin of the American Mathematical Society, 50(1):129–146, apr 2012.

References III

[Efr12b] Bradley Efron.
Bayesian inference and the parametric bootstrap.
Ann. Appl. Stat., 6(4):1971–1997, 12 2012.

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani.
The elements of statistical learning, volume 1.
Springer series in statistics New York, NY, USA:, 2001.

[GECd18] Alexandre Garcia, Slim Essid, Chloé Clavel, and Florence d'Alché-Buc.
Structured Output Learning with Abstention: Application to Accurate Opinion Prediction.
arXiv e-prints, page arXiv:1803.08355, March 2018.

References IV

[GG16] **Yarin Gal and Zoubin Ghahramani.**
Dropout as a bayesian approximation: Representing model uncertainty in deep learning.
In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1050–1059. JMLR.org, 2016.

[GPSW17] **Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger.**
On calibration of modern neural networks.
In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

References V

[GSC$^+$13] **Andrew Gelman, Hal S Stern, John B Carlin, David B Dunson, Aki Vehtari, and Donald B Rubin.**
Bayesian data analysis.
Chapman and Hall/CRC, 2013.

[HvC93] **Geoffrey E. Hinton and Drew van Camp.**
Keeping the neural networks simple by minimizing the description length of the weights.
In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, COLT '93, pages 5–13, New York, NY, USA, 1993. ACM.

References VI

[LAA+17] **Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl.**
Leveraging uncertainty information from deep neural networks for disease detection.
Scientific reports, 7(1):17816, 2017.

[LPB17] **Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell.**
Simple and scalable predictive uncertainty estimation using deep ensembles.
In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6402–6413. Curran Associates, Inc., 2017.

References VII

[MHB17] **Stephan Mandt, Matthew D. Hoffman, and David M. Blei.**
Stochastic gradient descent as approximate bayesian inference.
J. Mach. Learn. Res., 18(1):4873–4907, January 2017.

[Nea96] **Radford M. Neal.**
Bayesian Learning for Neural Networks.
Springer-Verlag, Berlin, Heidelberg, 1996.

[NPAA18] **Thejas Nair, Doina Precup, Douglas L. Arnold, and Tal Arbel.**
Exploring uncertainty measures in deep networks for multiple sclerosis lesion detection and segmentation.
In *MICCAI*, 2018.

References VIII

[RASC14] **Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson.**
Cnn features off-the-shelf: An astounding baseline for recognition.
In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '14, pages 512–519, Washington, DC, USA, 2014. IEEE Computer Society.

[RT96] **G. O. Roberts and R. L. Tweedie.**
Exponential convergence of Langevin distributions and their discrete approximations.
Bernoulli, 2(4):341–363, 1996.

[WJ08] **Martin J. Wainwright and Michael I. Jordan.**
Graphical models, exponential families, and variational inference.
Found. Trends Mach. Learn., 1(1-2):1–305, January 2008.

References IX

[WRV+20] Florian Wenzel, Kevin Roth, Bastiaan S. Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin.
How Good is the Bayes Posterior in Deep Neural Networks Really?
arXiv e-prints, page arXiv:2002.02405, February 2020.

[WT11] Max Welling and Yee Whye Teh.
Bayesian learning via stochastic gradient langevin dynamics.
In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 681–688, USA, 2011. Omnipress.

References X

[YCBL14] **Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson.**
How transferable are features in deep neural networks?
In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3320–3328, Cambridge, MA, USA, 2014. MIT Press.

[ZL17] **Lingxue Zhu and Nikolay Laptev.**
Deep and confident prediction for time series at uber.
2017 IEEE International Conference on Data Mining Workshops (ICDMW), pages 103–110, 2017.