TRADI: Tracking deep neural network weight distributions

Gianni Franchi[1,2] Andrei Bursuc[3] Emanuel Aldea[2] Séverine Dubuisson[4] Isabelle Bloch[5]

(1): ENSTA Paris, Institut polytechnique de Paris

(2): SATIE, Université Paris-Sud, Université Paris-Saclay

(3): valeo.ai

(4): CNRS, LIS, Aix Marseille University

(5): LTCI, Télécom Paris, Institut polytechnique de Paris

Narrator : Gianni FRANCHI

ECCV 2020 23-27/08/2020

Why do we study uncertainty with DNN?

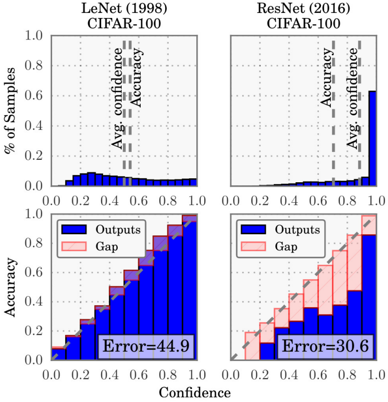
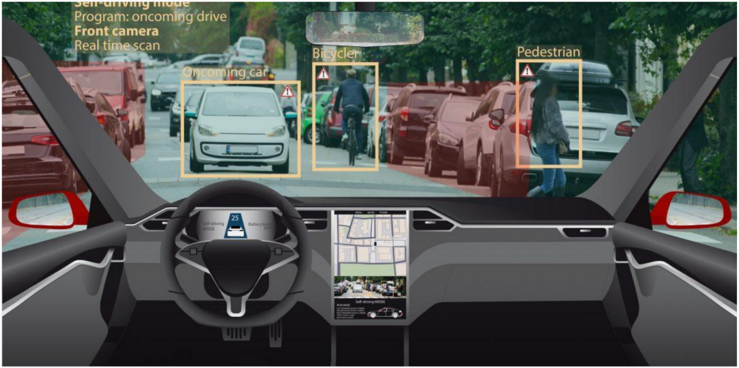


Figure: Confidence histograms (top) and reliability diagrams(bottom) for a 5-layer LeNet (left) and a 110-layer ResNet (right)on CIFAR-100. [1]

Why do we study uncertainty with DNN?

Imagine an autonomous car with a perception system based on Deep learning without Uncertainty:



Types of Uncertainty

- Aleatoric: Uncertainty inherent in the observation noise (problems caused by sensor quality, natural randomness, that cannot be explained by our data).
- Epistemic: Our ignorance about the correct model that generated the data (lack of knowledge about the process that generated the data).

Bayesian approach and DNN

The Goal of DNN is to find $\mathcal{P}(Y|X,\omega)$. In the classical bayesian approach we find ω such that we have the maximum a posteriori (MAP).

$$\hat{\omega} = \arg\max_{\omega} \log \mathcal{P}(\omega|\mathcal{D}_l)$$

$$\hat{\omega} = \arg\max_{\omega} \log \mathcal{P}(\mathcal{D}_l|\omega) + \log \mathcal{P}(\omega)$$

This leads to l2 regularization.

Bayesian DNN

Bayesian DNN is based on marginalization instead of MAP optimization.

$$\mathcal{P}(Y|X) = \mathbb{E}_{\omega \sim \mathcal{P}(\omega|\mathcal{D}_l)}\left(\mathcal{P}(Y|X,\omega)\right)$$
$$\mathcal{P}(Y|X) = \int \mathcal{P}(Y|X,\omega)\mathcal{P}(\omega|\mathcal{D}_l)d\omega$$

In practice:

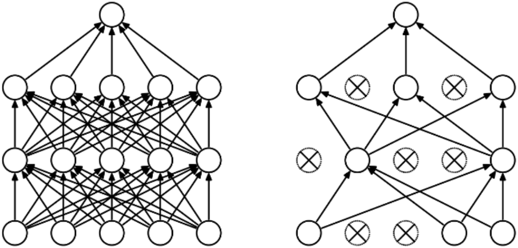
$$\mathcal{P}(Y|X) \simeq \sum_i \left(\mathcal{P}(Y|X,\omega_i)\right) \text{ with } \omega_i \sim \mathcal{P}(\omega|\mathcal{D}_l)$$

Different techniques to estimate $\mathcal{P}(\omega|\mathcal{D}_l)$.

Dropout[14]

Dropout is a technique that was proposed to avoid overfitting in CNN.
At each training step (i.e., for each sample within a mini-batch)

- Remove each node in the dropout layers with a probability p
- Update the weights of the remaining nodes with backpropagation.



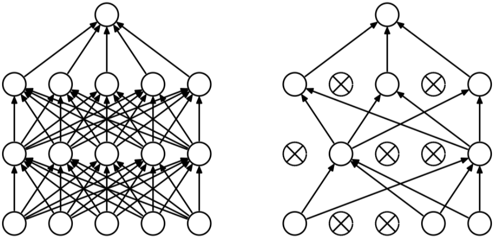
(a) Standard Neural Net (b) After applying dropout.

MC dropout [4]

Gal and Ghahramani [4] propose to average the predictions of several DNN where they apply the dropout:

$$\mathcal{P}(y^*|x^*) = \frac{1}{N_{\text{model}}} \sum_{j=1}^{N_{\text{model}}} \mathcal{P}(y^*|\omega(t^*) \odot b^j, x^*) \tag{1}$$

with b^j a vector of the same size of $\omega(t^*)$ which is a realization of a binomial distribution.

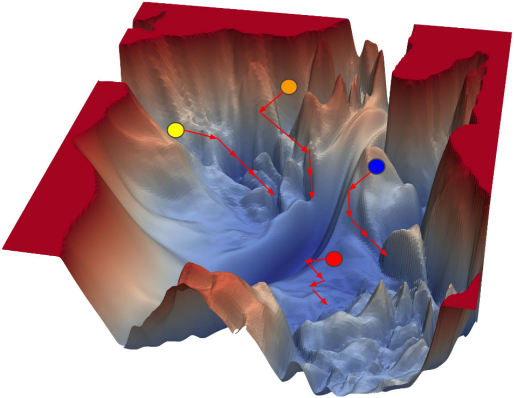


(a) Standard Neural Net (b) After applying dropout.

Deep Ensembles[5]

They [5] propose to average the predictions of several DNN with different initial seeds:

$$\mathcal{P}(y^*|x^*) = \frac{1}{N_{\text{model}}} \sum_{j=1}^{N_{\text{model}}} \mathcal{P}(y^*|\omega^j(t^*), x^*) \quad (2)$$



TRADI

- $\boldsymbol{\omega}(0)$ is the initial set of weights $\{\omega_k(0)\}_{k=1}^{K}$ following $\mathcal{N}(0, \sigma_k^2)$, where σ_k^2 are fixed as in [2].
- $\mathcal{L}(\boldsymbol{\omega}(t), y_i)$ is the loss function used to measure the dissimilarity between the output $g_{\boldsymbol{\omega}(t)}(x_i)$ of the DNN and the expected output y_i. One can use different loss functions.
- Weights on different layers are assumed to be independent of one another at all times. [9]
- Each weight $\omega_k(t)$, $k = 1, \ldots, K$, follows a non-stationary Normal distribution (e.g. $W_k(t) \sim \mathcal{N}(\mu_k(t), \sigma_k^2(t))$) whose two parameters are tracked.

TRADI

We had the following state and measurement equations for the mean $\mu_k(t)$:

$$\begin{cases} \mu_k(t) = \mu_k(t-1) - \eta \nabla \mathcal{L}_{\omega_k(t)} + \varepsilon_\mu \\ \omega_k(t) = \mu_k(t) + \tilde{\varepsilon}_\mu \end{cases} \tag{3}$$

with ε_μ being the state noise, and $\tilde{\varepsilon}_\mu$ being the observation noise, as realizations of $\mathcal{N}(0, \sigma_\mu^2)$ and $\mathcal{N}(0, \tilde{\sigma}_\mu^2)$ respectively.

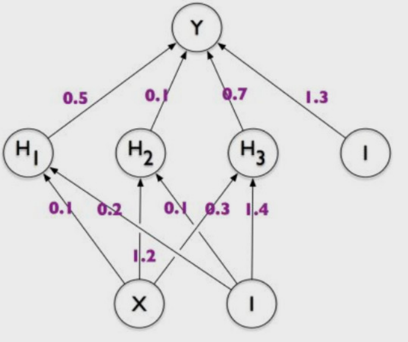
TRADI

The state and measurement equations for the variance σ_k are given by:

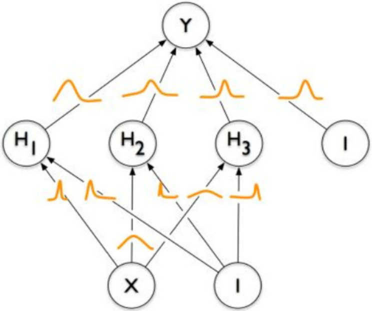
$$\begin{cases} \sigma_k^2(t) = \sigma_k^2(t-1) + \left(\eta \nabla \mathcal{L}_{\omega_{\boldsymbol{k}}(t)}\right)^2 + \varepsilon_\sigma \\ z_k(t) = \sigma_k^2(t) - \mu_k(t)^2 + \tilde{\varepsilon}_\sigma \\ \text{ with } z_k(t) = \omega_k(t)^2 \end{cases} \quad (4)$$

with ε_σ being the state noise, and $\tilde{\varepsilon}_\sigma$ being the observation noise, as realizations of $\mathcal{N}(0, \sigma_\sigma^2)$ and $\mathcal{N}(0, \tilde{\sigma}_\sigma^2)$, respectively.

TRADI



(Normal DNN)



(Bayesian DNN)

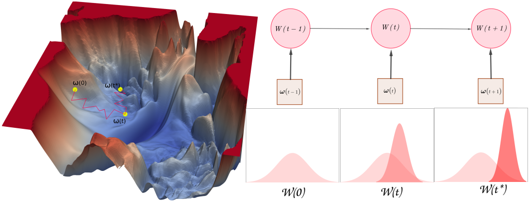
TRADI

We sample new realizations of $W(t*)$ using the following formula:

$$\tilde{\omega}(t^*) = \mu(t^*) + \Sigma^{1/2}(t^*) \times \mathrm{m}_1 \text{ with } \Sigma \text{ the covariance matrix.} \quad (5)$$

m_1 is a realization of the multivariate Gaussian $\mathcal{N}(0_K, \mathrm{I}_K)$. Then we take the expectation over this distribution :

$$\mathcal{P}(y^*|x^*) = \frac{1}{N_{\text{model}}} \sum_{j=1}^{N_{\text{model}}} \mathcal{P}(y^*|\tilde{\omega}^j(t^*), x^*) \quad (6)$$



Classification

Table: Comparative results on image classification

Method	MNIST		CIFAR-10	
	NLL	ACCU	NLL	ACCU
Deep Ensembles	**0.035**	**98.88**	0.173	95.67
MC Dropout	0.065	98.19	0.205	95.27
SWAG	0.041	98.78	**0.110**	**96.41**
TRADI (ours)	0.044	98.63	0.205	95.29

Metrics[1]

First we group predictions into M bins, each of size $1/M$. Let B_m be the set of indices of samples whose prediction confidence falls into the interval $I_m =]m-1/M, m/M]$.
The accuracy of a set B_m is defined as:

$$\mathrm{acc}(B_m) = 1/|B_m| \sum_{i \in B_m} \delta_{y_i}(\hat{y}_i) \quad (7)$$

The average confidence in B_m is defined as:

$$\mathrm{conf}(B_m) = 1/|B_m| \sum_{i \in B_m} \hat{p}_i \quad (8)$$

where $\hat{p}_i$ is the confidence for sample i.

Metrics [1]

Expected Calibration Error (ECE) measures the difference in expected accuracy and expected confidence. It is defined as:

$$ECE = \sum_{m}^{M} 1/|B_m||\mathrm{acc}(B_m) - \mathrm{conf}(B_m)| \quad (9)$$

Metrics[11]

The dataset is divided in two:

- Out of distribution
- in distribution

The confidence score $\hat{p}_i$ for sample i $\hat{p}_i$ is used to detect OOD data. To eveluate the quality we can use :

- Area Under the ROC Curve → AUC
- Area Under the Average Precision Curve → AUPR
- FPR at 95% TPR can be interpreted as the probability that a negative (out-of-distribution)example is misclassified as positive (in-distribution) when the true positive rate (TPR) is as high as 95%. True positive rate can be computed by TPR = TP / (TP+FN) and , the false positive rate (FPR) can be computed by FPR =FP / (FP+TN).

Out of distribution (Results on the CamVid experiments)

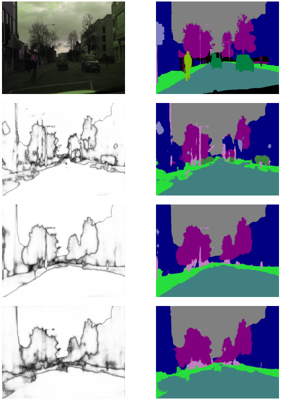
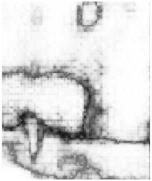


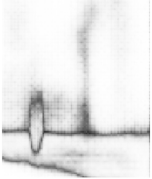
Figure: First row: input image and ground truth, second, third and fourth rows: output and confidence score given by MC dropout, Deep Ensembles and our TRADI, respectively.

Out of distribution

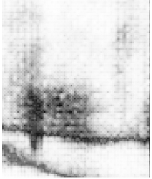
(a) input image



(b) MC dropout confidence



(c) Deep Ensembles confidence



(d) TRADI confidence

Figure: Zooms of the confidence results on the CamVid experiments. In the bottom left of the input image (a), there is a human, hence a pixel region of an unknown class for all the DNNs, since the pedestrian class was amongst the ones marked as unlabeled. Yet, only the TRADI DNN (d) is consistent.

Out of distribution

Dataset	OOD technique	AUC	AUPR	FPR-95%-TPR	ECE	Train time
MNIST/notMNIST 3 hidden layers	Baseline (MCP)	94.0	96.0	24.6	0.305	2m
	Gauss. perturbation ensemble	94.8	96.4	19.2	0.500	2m
	MC Dropout	91.8	94.9	35.6	0.494	2m
	Deep Ensembles	97.2	98.0	9.2	0.462	31m
	TRADI (ours)	96.7	97.6	11.0	0.407	2m
CamVid-OOD ENET	Baseline (MCP)	75.4	10.0	65.1	0.146	30m
	Gauss. perturbation ensemble	76.2	10.9	62.6	0.133	30m
	MC Dropout	75.4	10.7	63.2	0.168	30m
	Deep Ensembles	79.7	13.0	55.3	0.112	5h
	TRADI (ours)	79.3	12.8	57.7	0.110	41m
StreetHazards PSPNet	Baseline (MCP)	88.7	6.9	26.9	0.055	13h14m
	Gauss. perturbation ensemble	57.08	2.4	71.0	0.185	13h14m
	MC Dropout	69.9	6.0	32.0	0.092	13h14m
	Deep Ensembles	90.0	7.2	25.4	0.051	132h19m
	TRADI (ours)	89.2	7.2	25.3	0.049	15h36m
BDD Anomaly PSPNet	Baseline (MCP)	86.0	5.4	27.7	0.159	18h08
	Gauss. perturbation ensemble	86.0	4.8	27.7	0.158	18h08m
	MC Dropout	85.2	5.0	29.3	0.181	18h08m
	Deep Ensembles	87.0	6.0	25.0	0.170	189h40m
	TRADI (ours)	86.1	5.6	26.9	0.157	21h48m

Bibliography:

1 Guo, Chuan, et al. "On calibration of modern neural networks." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017.

2 He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." Proceedings of the IEEE international conference on computer vision. 2015.

3 Li, Hao, et al. "Visualizing the loss landscape of neural nets." Advances in Neural Information Processing Systems. 2018.

4 Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." international conference on machine learning. 2016.

5 Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles." Advances in neural information processing systems. 2017.

Bibliography:

6 Kendall, Alex, and Yarin Gal. "What uncertainties do we need in bayesian deep learning for computer vision?." Advances in neural information processing systems. 2017.

7 Yoo, Donggeun, and In So Kweon. "Learning loss for active learning." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.

8 Corbiere, Charles, et al. "Addressing Failure Prediction by Learning Model Confidence." Advances in Neural Information Processing Systems. 2019.

9 Blundell, Charles, et al. "Weight uncertainty in neural networks." arXiv preprint arXiv:1505.05424 (2015).

10 Fort, Stanislav, Huiyi Hu, and Balaji Lakshminarayanan. "Deep Ensembles: A Loss Landscape Perspective." arXiv preprint arXiv:1912.02757 (2019).

11 Hendrycks, Dan, et al. "A Benchmark for Anomaly Segmentation." arXiv preprint arXiv:1911.11132 (2019).

Bibliography:

12 Bishop, Christopher M. "Mixture density networks." (1994).

13 Liao et al, Defense against Adversarial Attacks Using High-Level Representation Guided Denoiser, 2017

14 Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.