TRADI: Tracking deep neural network weight distributions

Gianni FRANCHI[†]
In collaboration with: Emanuel Aldea[Δ], Andrei Bursuc[*], Severine Dubuisson[◇], Isabelle Bloch[•]

† : *U2IS, ENSTA Paris*, Δ : SATIE, Université Paris-Saclay, ∗ : valeo.ai, ◇ : CNRS, LIS, • : LTCI, Télécom Paris

GDR ISIS 2020
11/05/2020

Plan

What is uncertainty in machine/deep learning

- We make observations using the sensors in the world (e.g. camera)
- Based on the observations, we intend to learn a model that makes decisions
- Given the **same** observations, the decision should be the **same**

However,

- The world **changes**, observations **change**, our sensors **change**, the output should not change!
- We would like to know how confident we can be about the decisions

Why Uncertainty is important?

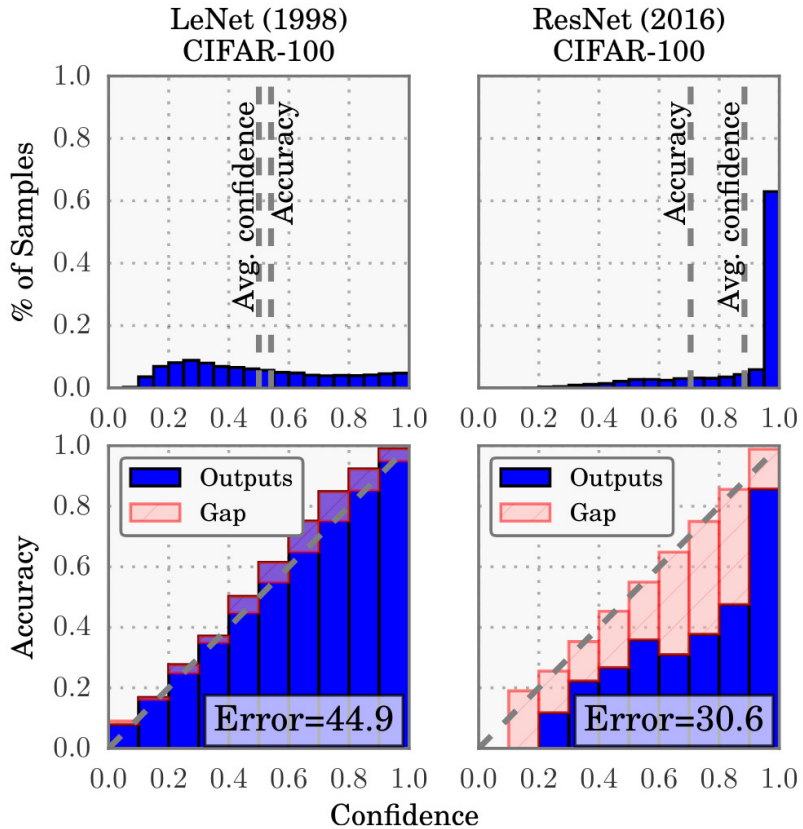
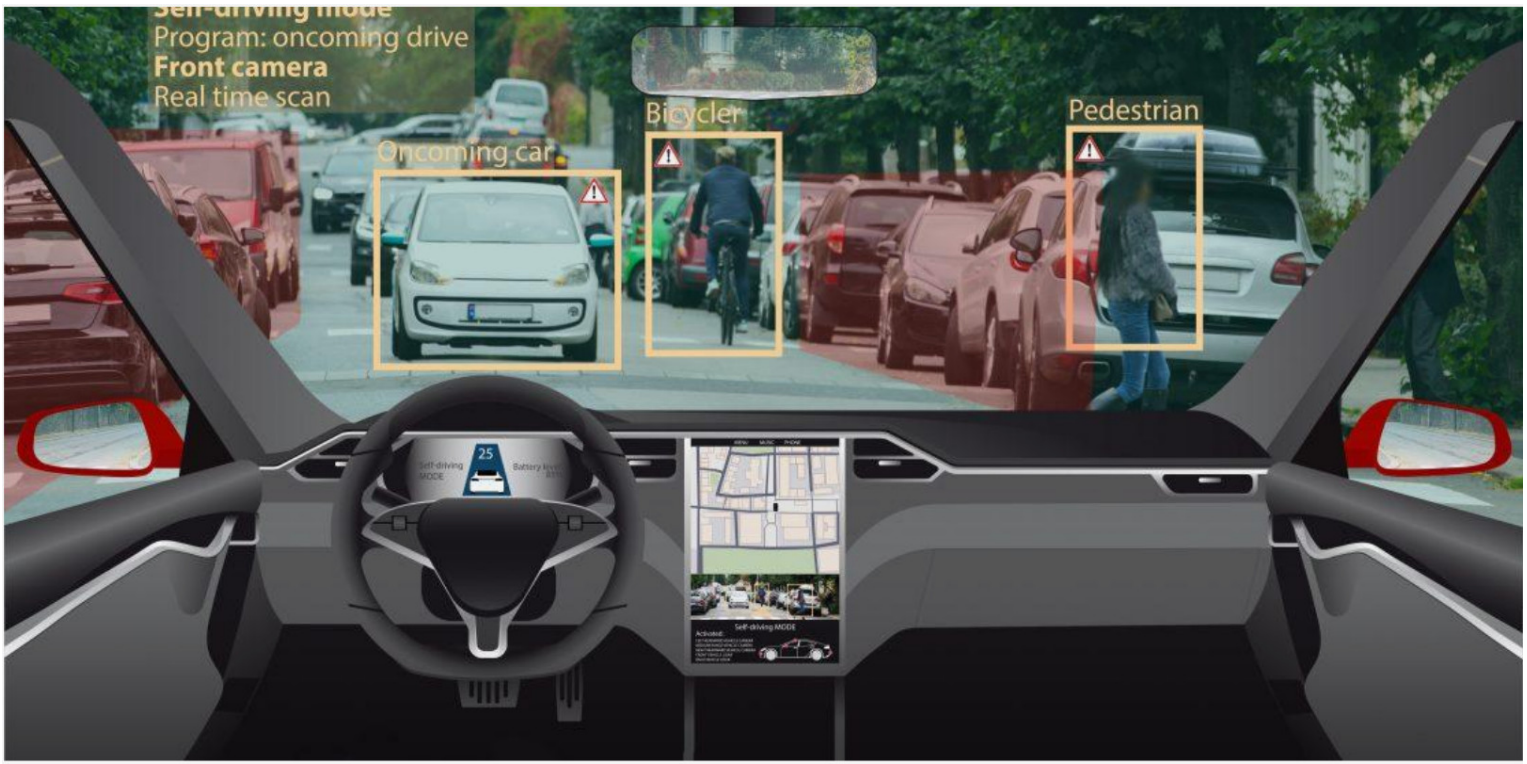


Figure: Confidence histograms (top) and reliability diagrams(bottom) for a 5-layer LeNet (left) and a 110-layer ResNet (right)on CIFAR-100. [1]

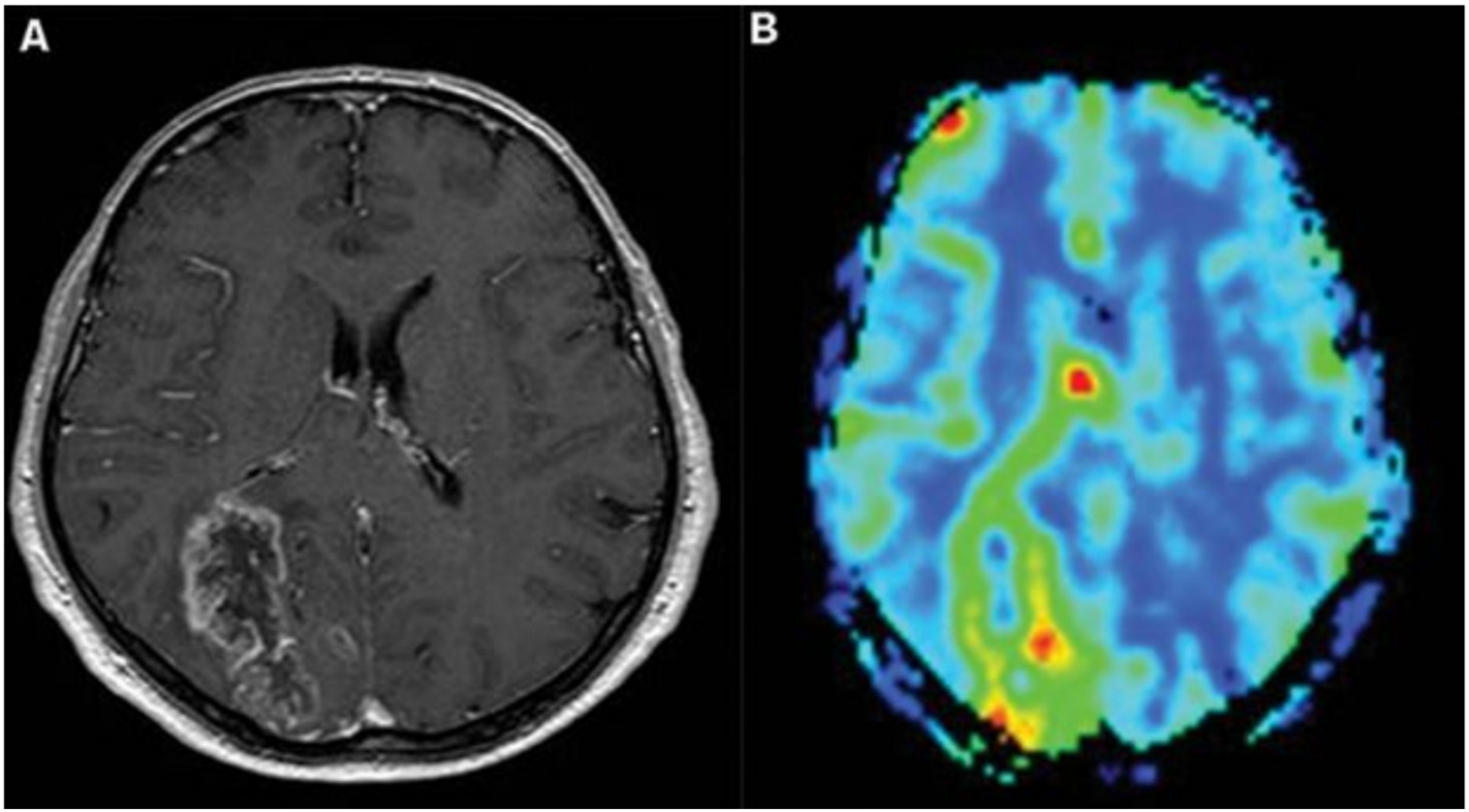
Why Uncertainty is important?

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



Why Uncertainty is important?

Imagine a medical diagnostics based on Deep learning without Uncertainty:

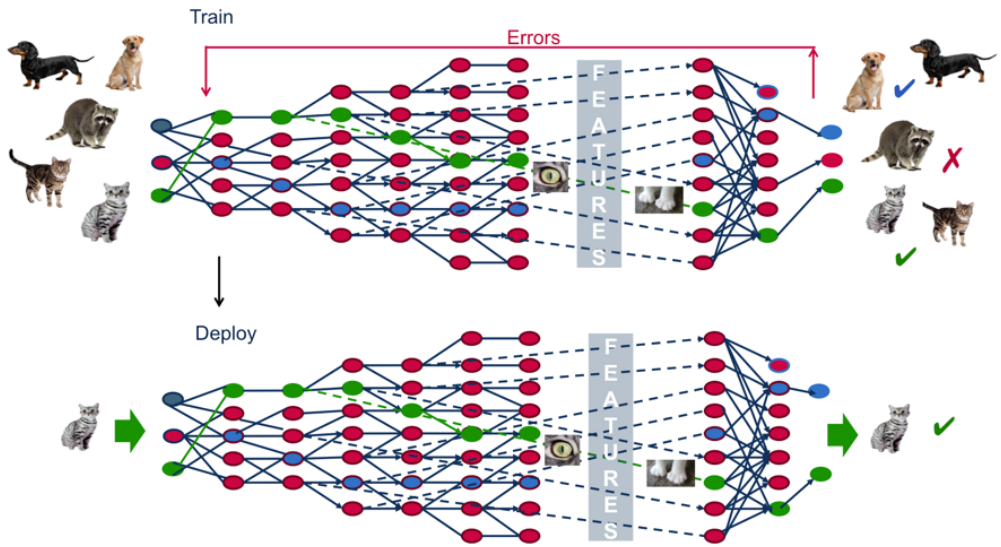


Why Uncertainty is important?

We build models for predictions, can we trust them? Are they certain?

Deep Learning

Deep learning systems are neural network (or convolutional neural network) models similar to those popular in the '80s and '90s, with algorithmic inovations, software inovations, and larger data sets.

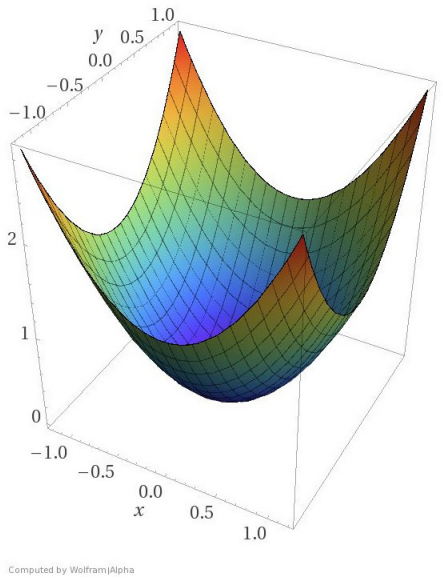


Deep Learning notations

- Training/Testing sets are denoted respectively by $\mathcal{D}_l = (x_i, y_i)_{i=1}^{n_l}$, $\mathcal{D}_\tau = (x_i, y_i)_{i=1}^{n_\tau}$. Without loss of generality we consider the observed samples $\{x_i\}_{i=1}^{n}$ and the corresponding labels $\{y_i\}_{i=1}^{n}$ as vectors. Data in $\mathcal{D}_l$ and $\mathcal{D}_\tau$ are assumed to be i.i.d. distributed according to their respective unknown joint distribution $\mathcal{P}_t$ and $\mathcal{P}_\tau$.
- The Deep Neural Networks (DNN) are function parameterized by a vector containing the K trainable weights $\boldsymbol{\omega} = \{\omega_k\}_{k=1}^{K}$.
- During training, $\boldsymbol{\omega}$, is iteratively updated for each mini-batch and we denote by $\boldsymbol{\omega}(t)$ the state of the DNN at iteration t of the optimization algorithm, and following the random variable $W(t)$.
- g represents the architecture of the DNN associated with these weights and $g_{\boldsymbol{\omega}(t)}(x_i)$ its output at t.

Deep Learning optimization

We denote: $\mathcal{L}(\omega(t), y_i)$ the loss function used to measure the dissimilarity between the output $g_{\omega(t)}(x_i)$ of the DNN and the expected output y_i. One can use different loss functions.
We use can use a gradient descent to optimize $\omega(t)$



However, for large neural networks with a large training set, computing the gradient is costly, and the loss is not convex.

Deep Learning optimization

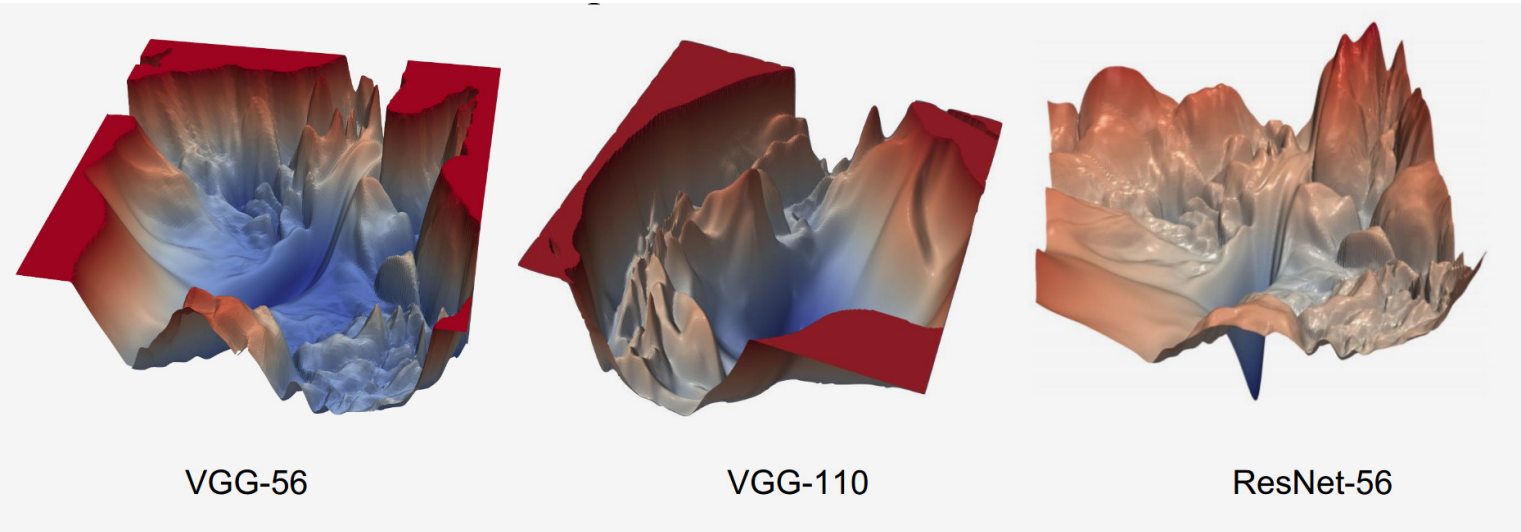


Figure: Visualizing the loss surfaces of modern DNN [3]

Deep Learning optimization

For that, we consider **stochastic gradient descent** SGD algorithm on a mini-batch in order to optimize the loss between two weight realizations. The loss derivative with respect to a given weight $\omega_k(t)$ on a mini-batch $B(t)$ is given by:

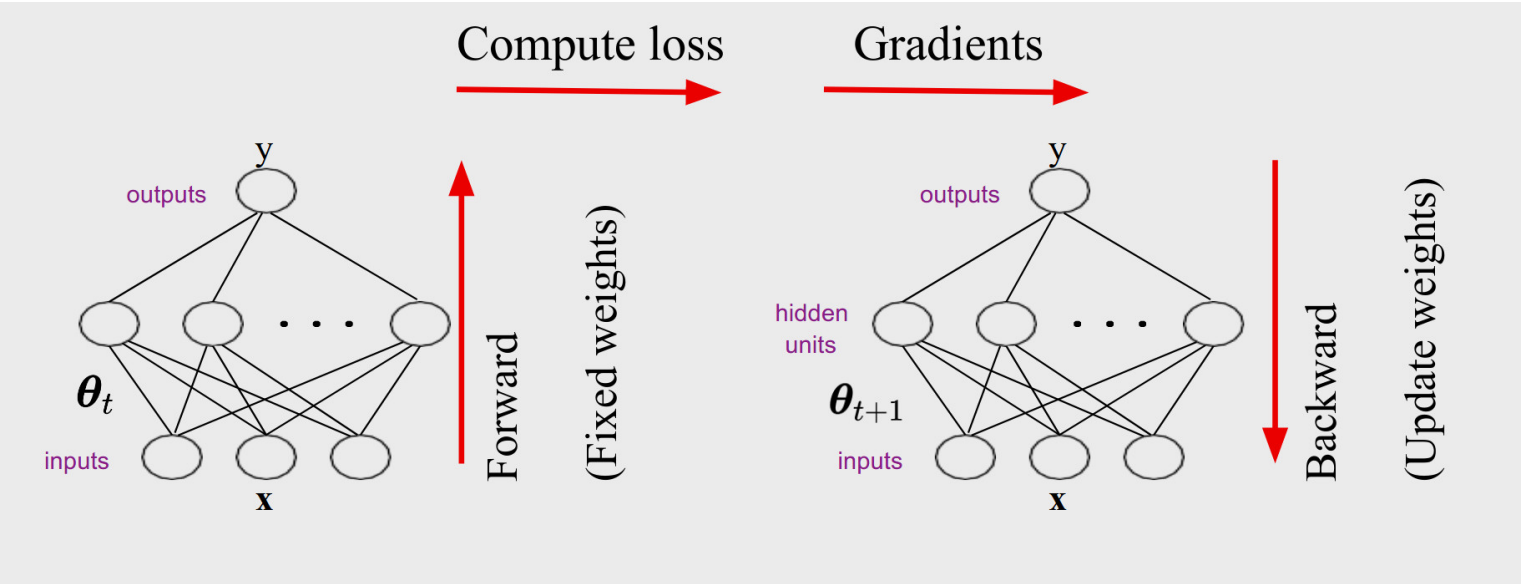
$$\nabla \mathcal{L}_{\omega_k}(t) = \frac{1}{|B(t)|} \sum_{(x_i, y_i) \in B(t)} \frac{\partial \mathcal{L}(\boldsymbol{\omega}(t-1), y_i)}{\partial \omega_k(t-1)} \tag{1}$$

Weights $\omega_k(t)$ are then updated as follows:

$$\omega_k(t) = \omega_k(t-1) - \eta \nabla \mathcal{L}_{\omega_k}(t) \tag{2}$$

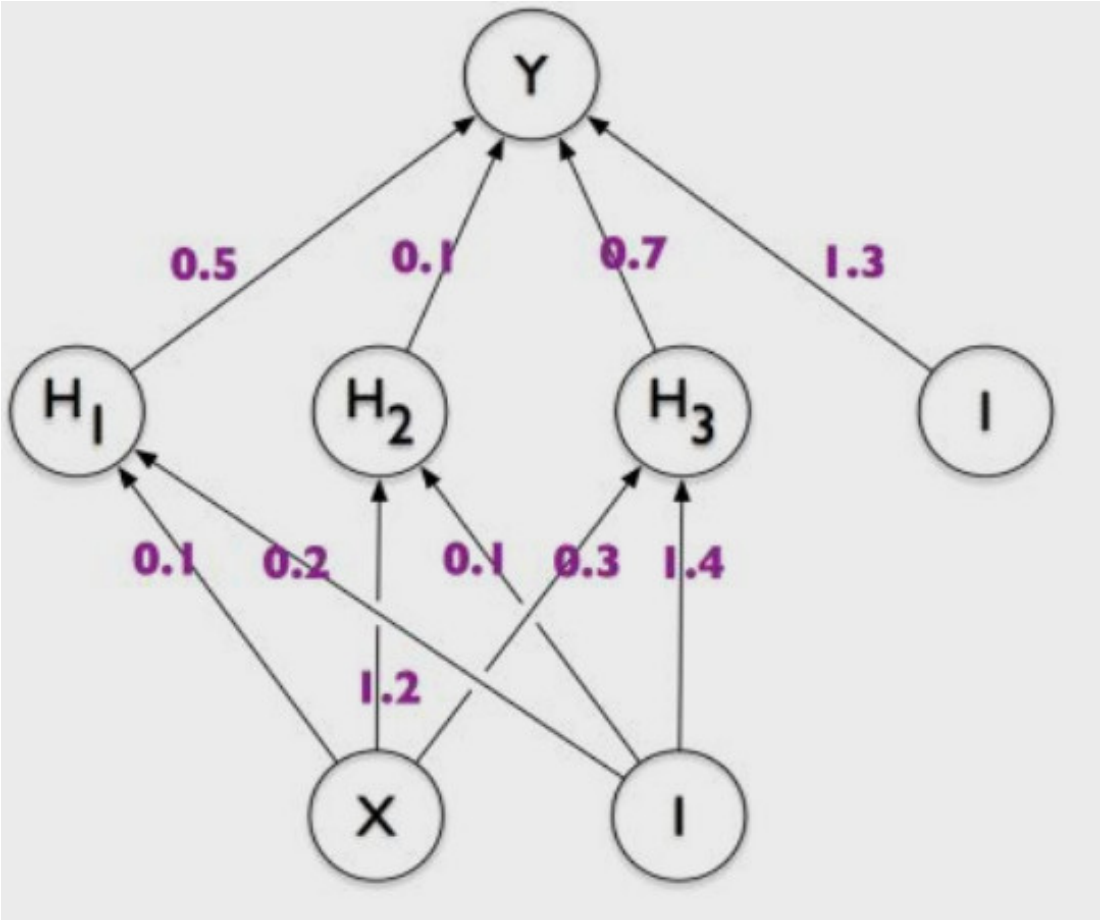
with η the learning rate.

Deep Learning optimization



Deep Learning testing

When we test a DNN on new data we just test with the optimal $\omega(t^*)$ (one realisation $W(t^*)$)



Types of Uncertainty

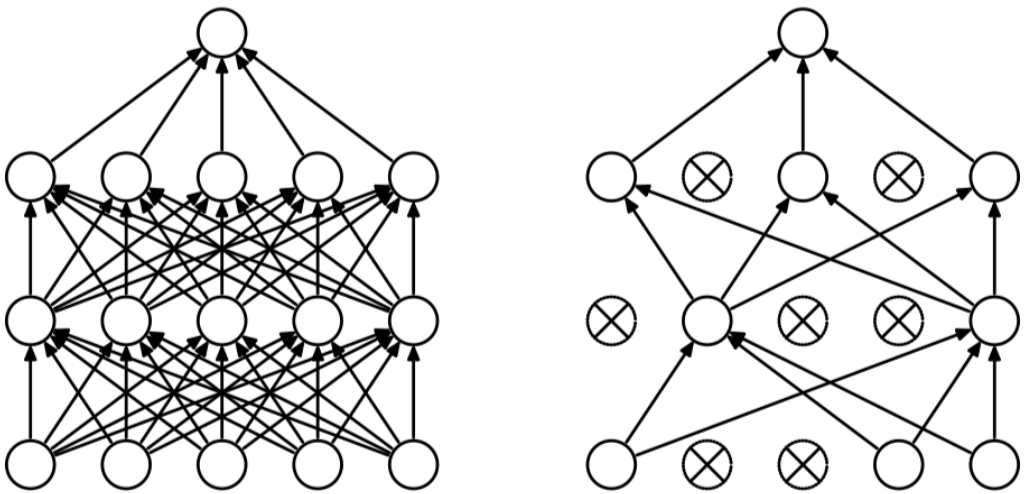
- Aleatoric: Uncertainty inherent in the observation noise (problems caused by sensor quality, natural randomness, that cannot be explained by our data).
- Epistemic: Our ignorance about the correct model that generated the data (lack of knowledge about the process that generated the data).

Current solutions : MC dropout [4]

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

$$\mathcal{P}(y^*|x^*) = \frac{1}{N_{\text{model}}} \sum_{j=1}^{N_{\text{model}}} \mathcal{P}(y^*|\omega(t^*) \odot b^j, x^*) \tag{3}$$

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

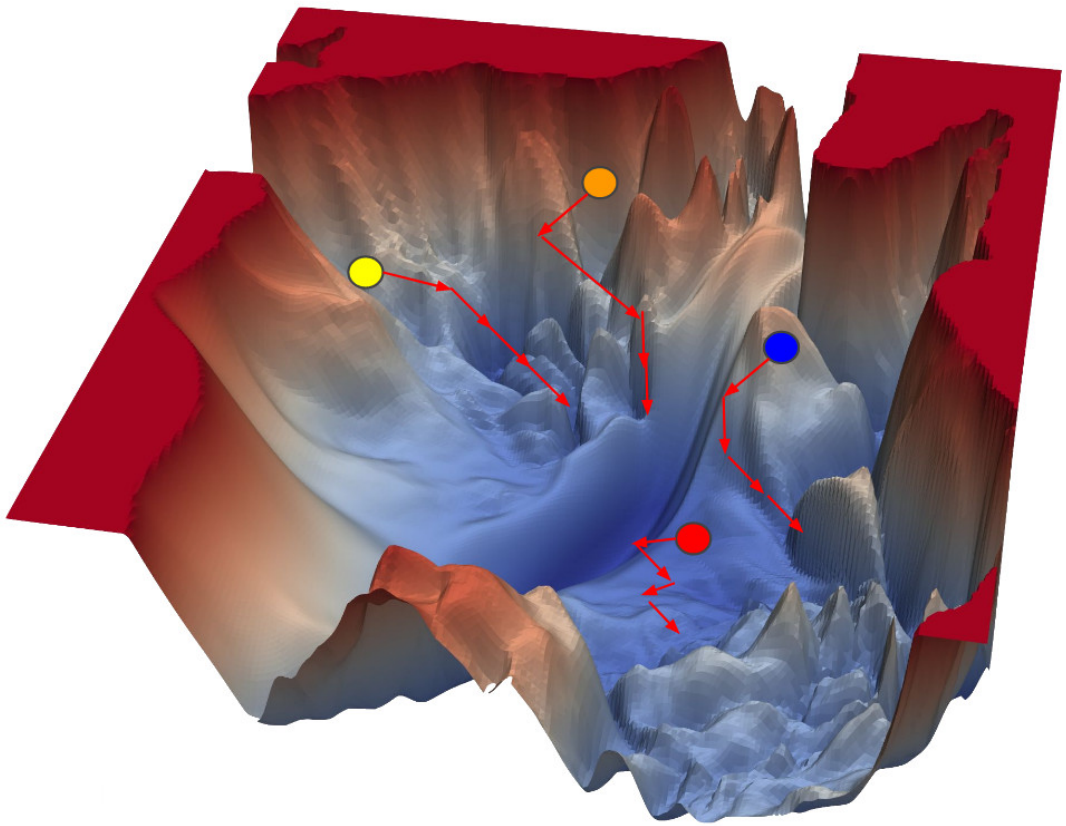


(a) Standard Neural Net (b) After applying dropout.

Current solutions : Deep Ensembles[5]

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

$$\mathcal{P}(y^*|x^*) = \frac{1}{N_{\text{model}}} \sum_{j=1}^{N_{\text{model}}} \mathcal{P}(y^*|\omega^j(t^*), x^*) \tag{4}$$



Deep Learning our proposal

- $\boldsymbol{\omega}(0)$ 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.
- 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.

Deep Learning TRADI(our proposal)

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

$$\begin{cases} \mu_k(t) = \mu_k(t-1) - \eta \nabla \mathcal{L}_{\omega_k(t)} + \varepsilon_\mu \\ \omega_k(t) = \mu_k(t) + \tilde{\varepsilon}_\mu \end{cases} \tag{5}$$

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.

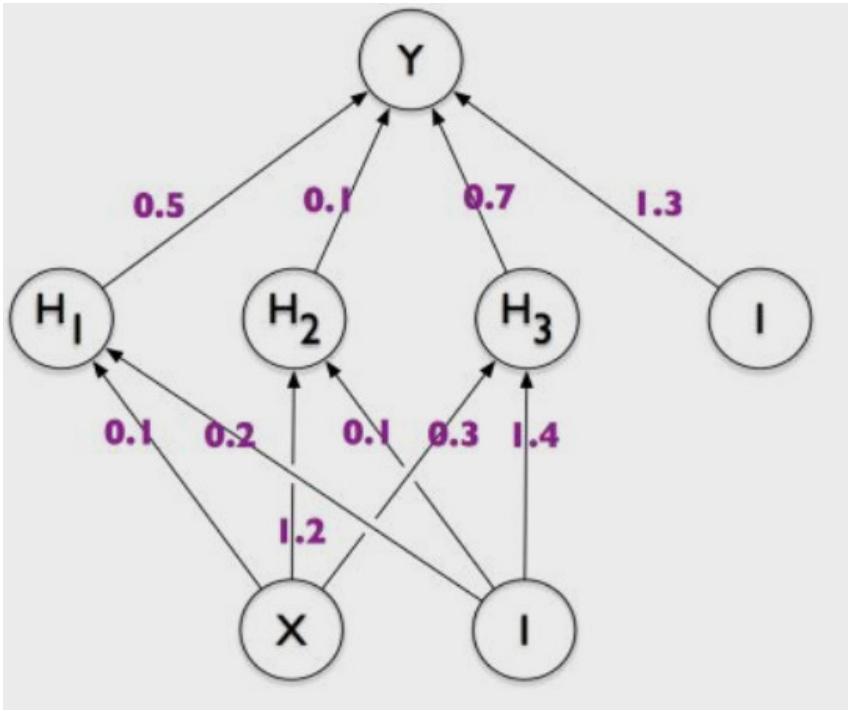
Deep Learning TRADI(our proposal)

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

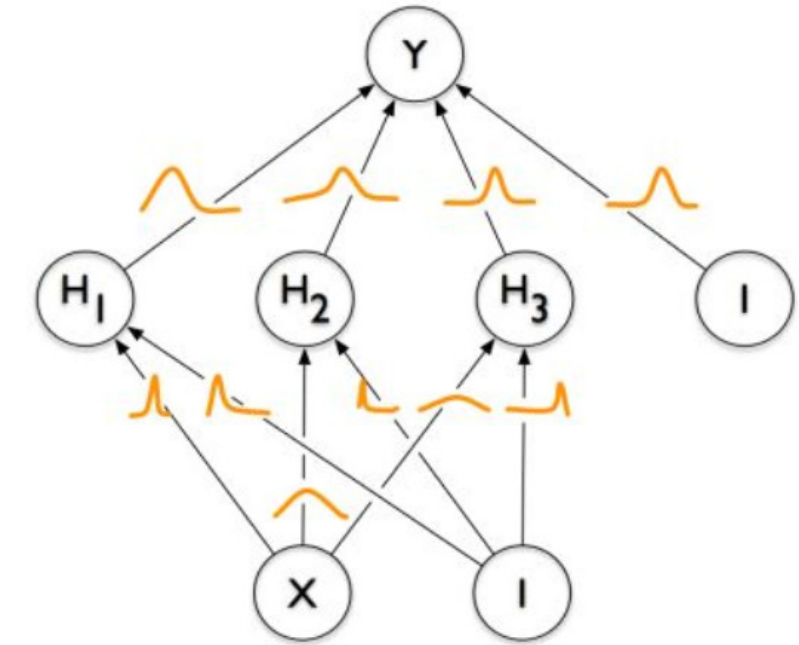
$$\begin{cases} \sigma_k^2(t) = \sigma_k^2(t-1) + \left(\eta \nabla \mathcal{L}_{\omega_k(t)}\right)^2 - \eta^2 \mu_k(t)^2 + \varepsilon_\sigma \\ z_k(t) = \sigma_k^2(t) + \mu_k(t)^2 + \tilde{\varepsilon}_\sigma \\ \text{with } z_k(t) = \omega_k(t)^2 \end{cases} \tag{6}$$

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.

Current solutions : TRADI(our proposal)



(Normal DNN)



(Bayesian DNN)

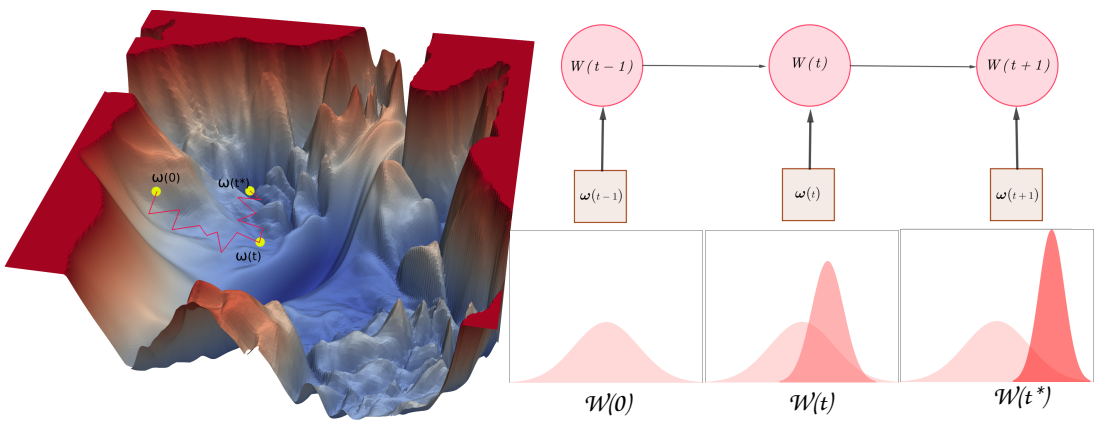
Current solutions : TRADI(our proposal)

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

$$\tilde{\omega}(t^*) = \mu(t^*) + \Sigma^{1/2}(t^*) \times \boldsymbol{m}_1 \text{ with } \Sigma \text{ the covariance matrix.} \quad (7)$$

$\boldsymbol{m}_1$ is a realization of the multivariate Gaussian $\mathcal{N}(\mathbf{0}_K, \boldsymbol{I}_K)$. Then we take the expectation over this distribution :

$$\mathcal{P}(y^*|x^*) = \frac{1}{N_{\text{model}}} \sum_{j=1}^{N_{\text{model}}} \mathcal{P}(y^*|\tilde{\omega}^j(t^*), x^*) \quad (8)$$



Regression

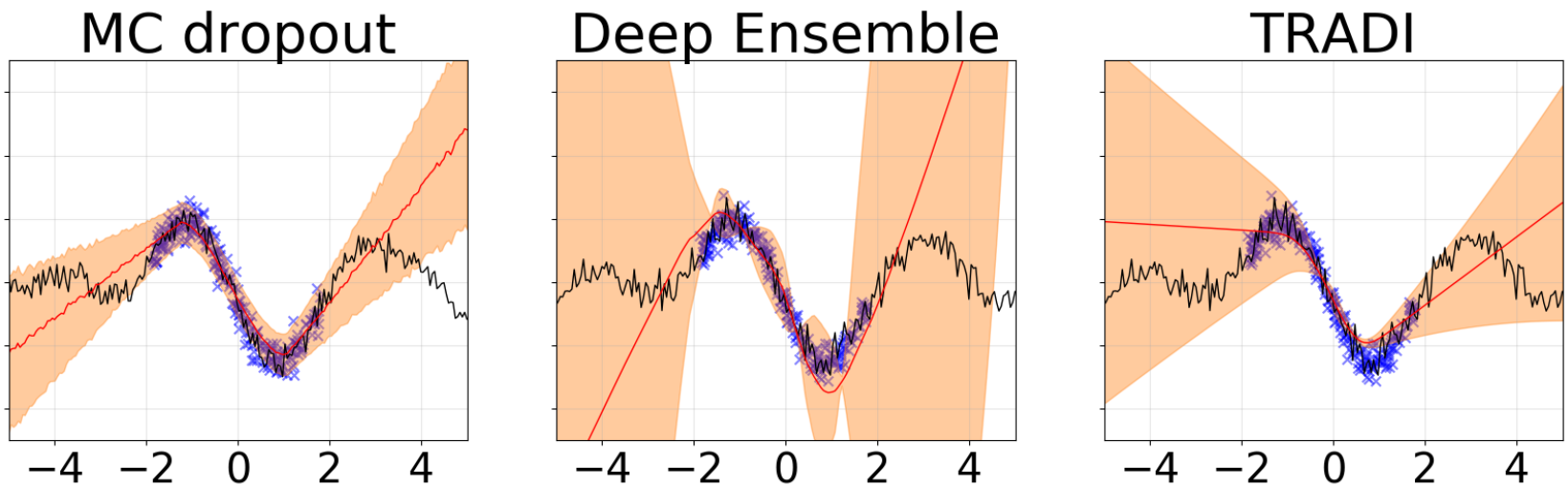


Figure: Results on a synthetic regression task with MC dropout, Deep Ensembles and TRADI algorithm. x-axis: spatial coordinate of the Gaussian process. Black lines: ground truth curve. Orange areas: estimated variance. Blue points represents the training points.

Classification

Table: Comparative results on image classification

Method	MNIST		CIFAR-10	
	NLL	ACCU	NLL	ACCU
Deep Ensembles [5]	**0.035**	**98.88**	**0.173**	**95.67**
MC Dropout [4]	0.065	98.19	0.205	95.27
TRADI	0.044	98.63	0.205	95.29

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
	MC Dropout [4]	91.8	94.9	35.6	0.494	2m
	Deep Ensemble [5]	97.2	98.0	9.2	0.462	31m
	OVNNI (ours)	99.3	99.6	3.5	0.066	...
	ODIN [7]	94.9	96.7	17.5	0.500	...
	ConfidNet [6]	97.9	99.0	12.7	0.461	...
	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
	MC Dropout [4]	75.4	10.7	63.2	0.168	30m
	Deep Ensemble [5]	79.7	13.0	55.3	0.112	5h
	OVNNI (ours)	96.1	61.2	16.5	0.025	...
	ConfidNet [6]	81.9	13.8	55.8	0.121	...
	TRADI (ours)	79.3	12.8	57.7	0.110	41m

Out of distribution

Dataset	OOD technique	AUC	AUPR	FPR-95%-TPR	ECE	Train time
StreetHazards PSPNet	Baseline (MCP)	88.7	6.9	26.9	0.055	13h14m
	MC Dropout [4]	69.9	6.0	32.0	0.092	13h14m
	Deep Ensemble [5]	90.0	7.2	25.4	0.051	132h19m
	OVNNI (ours)	91.2	12.6	22.2	0.048	...
	ConfidNet [6]	83.6	2.3	26.2	0.10	...
	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
	MC Dropout [4]	85.2	5.0	29.3	0.181	18h08m
	Deep Ensemble [5]	87.0	6.0	25.0	0.170	189h40m
	OVNNI (ours)	87.2	6.7	25.0	0.081	...
	ConfidNet [6]	85.4	5.1	29.1	0.232	...
	TRADI (ours)	86.1	5.6	26.9	0.157	21h48m

Out of distribution

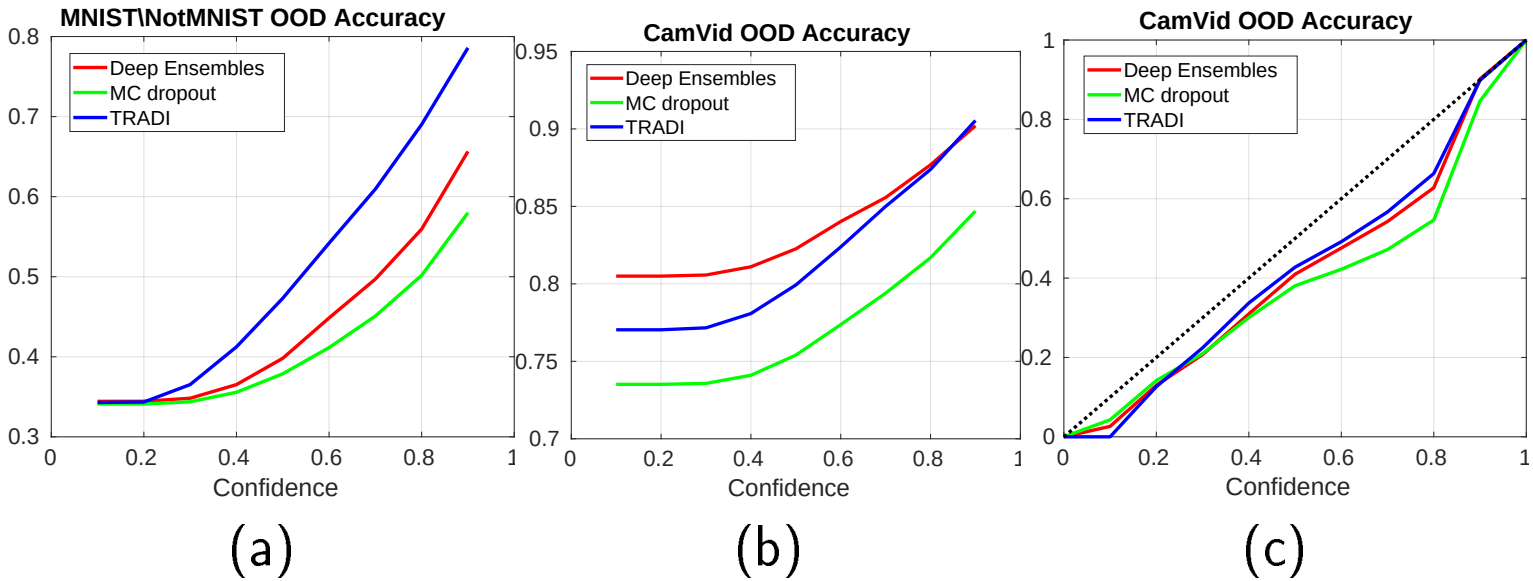


Figure: (a) and (b) Accuracy vs confidence plot on the MNIST \NotMNIST and CamVid experiments, respectively. (c) Calibration plot for the CamVid experiment.

Out of distribution (Results on the CamVid experiments)

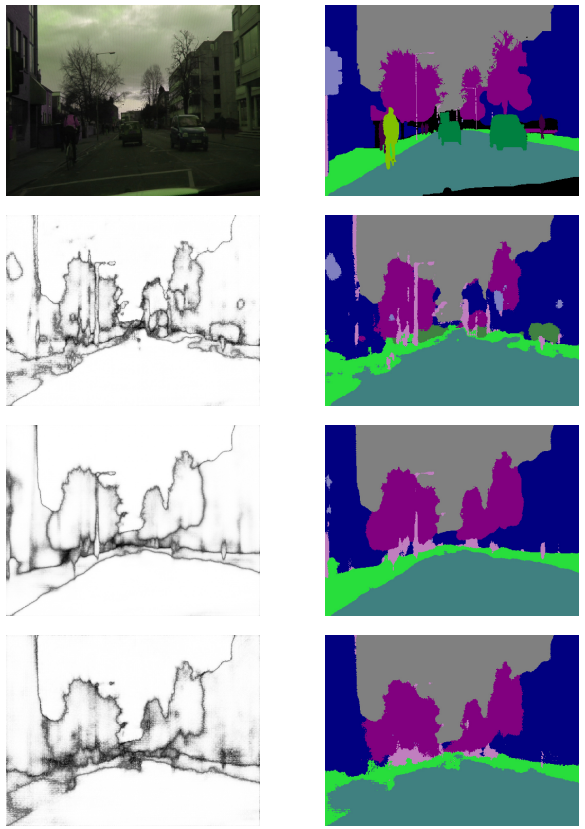
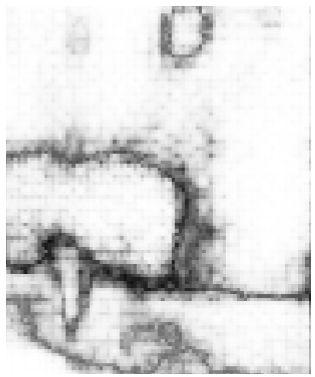


Figure: First row: input image and ground truth, second, third and fourth rows: output and confidence score given by MC dropout, Deep Ensembles and our TRADI, respectively.

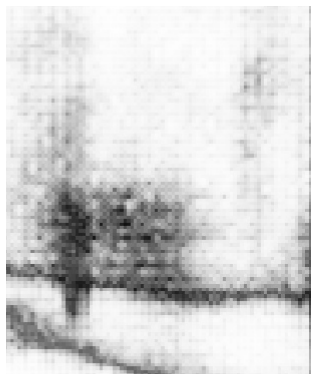
Out of distribution

(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.

Conclusions:

We have conducted a study that allow us to have access to the distribution of the DNN.
We have access to the distribution of the DNN for almost free.
We have tested our Bayesian DNN on different datasets and have good results.
Thanks for your attention.

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.

Bibliography:

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

6 Corbi¨¨re, Charles, et al. "Addressing Failure Prediction by Learning Model Confidence." Advances in Neural Information Processing Systems. 2019.

7 Liang, Shiyu, Yixuan Li, and Rayadurgam Srikant. "Enhancing the reliability of out-of-distribution image detection in neural networks." arXiv preprint arXiv:1706.02690 (2017).