TRADI: Tracking deep neural network weight distributions (Supplementary material)

Gianni Franchi[1,2], Andrei Bursuc[3], Emanuel Aldea[2], Séverine Dubuisson[4], and 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

1 TRAcking of the DIstribution (TRADI) of weights of a neural network

This section details the tracking of the covariance matrix of a shallow neural network (see Section 3.2.1 of the main paper), gives the whole TRADI algorithm and explains how its parameters are chosen.

1.1 Kalman Filtering

Bayesian recursive filtering [3] aims to estimate the state of a hidden Markov process, observed through a state space equations system. Let $\{\mu_k(0), ..., \mu_k(t)\}$ be this process, where $\mu_k(t)$ is the state, and $\omega_k(t)$ the observation. At time t the filtering equations in the absence of a control model are:

$$\begin{cases} \mu_k(t) = f_t(\mu_k(t-1)) \\ \omega_k(t) = g_t(\mu_k(t)) \end{cases} \tag{1}$$

The first equation is the state equation, with f_t the transition function between times $t-1$ and t and the second is the observation equation, with g_t the measurement function. Under the Markov assumptions, the probability of the current state given the immediately previous one is conditionally independent of the other earlier states, in addition, the measurement at the kth timestep is only dependent on the current state. Moreover, the Kalman model considers that these two equations are perturbed by white Gaussian noises which are independent from each other. Finally, if f_t and g_t are linear operators, than the Kalman filter is an optimal estimator.

1.2 Tracking the mean and variance of the weights and the covariance

We consider a neural network (NN) for which each layer has few neurons, less than 100. Our goal here is to estimate, for all weights $\omega_k(t)$ of the NN and at each time step t of the training process, $\mu_k(t)$ and $\sigma_k^2(t)$ the parameters of their normal distribution. Furthermore, we want to estimate $\Sigma_{k,k'}(t)$ which is the covariance matrix between

the $\omega_k(t)$ and $\omega_{k'}(t)$. Note that, since we assume that weights on different layers are independent, we evaluate the covariance for $k, k^{\prime}$ belonging to the same layer, otherwise their covariance is null. To this end, we leverage mini-batch SGD to optimize the loss between two weight realizations.

The derivative of the loss with respect to a given weight $\omega_k(t-1)$ over a mini-batch $B(t)$ is given by:

$$\nabla \mathcal{L}_{\omega_k(t)} = \frac{1}{|B(t)|} \sum_{(x_i, y_i) 2 B(t)} \frac{\partial \mathcal{L}(\, !\, (t-1), y_i)}{\partial \omega_k(t-1)} \tag{2}$$

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

$$\omega_k(t) = \omega_k(t-1) - \eta \nabla \mathcal{L}_{\omega_k(t)} \tag{3}$$

The weights of NNs are randomly initialized at $t = 0$ by sampling $W_k(0) \sim \mathcal{N}(\mu_k(0), \sigma_k^2(0))$, where the parameters of the distribution are set empirically on a per-layer basis as in [2]. In addition, for all couples of weights $(k, k^{\prime})$, the corresponding element of the covariance matrix is given by $\Sigma_{k,k'}(0) = 0$ since all the weights are considered independent at time $t = 0$.

Similarly with the main article, we use the following state and measurement equations for the mean $\mu_k(t)$:

$$\begin{cases} \mu_k(t) = \mu_k(t-1) - \eta \nabla \mathcal{L}_{\omega_k(t)} + n_\mu \\ \omega_k(t) = \mu_k(t) + \tilde{n}_\mu \end{cases} \tag{4}$$

where n_μ is the state noise, and $\tilde{n}_\mu$ the observation noise, realizations of $\mathcal{N}(0, \sigma_\mu^2)$ and $\mathcal{N}(0, \tilde{\sigma}_\mu^2)$ respectively. The state and measurement equations for the variance σ_k are given by:

$$\begin{cases} \sigma_k^2(t) = \sigma_k^2(t-1) + \left(\eta \nabla \mathcal{L}_{\omega_k(t)}\right)^2 + n_\sigma \\ z_k(t) = \sigma_k^2(t) - \mu_k(t)^2 + \tilde{n}_\sigma \\ \text{with } z_k(t) = \omega_k(t)^2 \end{cases} \tag{5}$$

where n_σ is the state noise, and $\tilde{n}_\sigma$ is the observation noise, realizations of $\mathcal{N}(0, \sigma_\sigma^2)$ and $\mathcal{N}(0, \tilde{\sigma}_\sigma^2)$ respectively. As proposed in the main article, and similarly with [1, 6], we assume that weights during back-propagation and forward pass are independent. We then get:

$$\begin{aligned} \Sigma(t)_{k,k'} = \Sigma(t-1)_{k,k'} + \\ \eta^2 \mathbb{E}\left[\nabla \mathcal{L}_{\omega_k(t)} \nabla \mathcal{L}_{\omega_{k'}(t)}\right] - \eta^2 \mathbb{E}\left[\nabla \mathcal{L}_{\omega_k(t)}\right] \mathbb{E}\left[\nabla \mathcal{L}_{\omega_{k'}(t)}\right] \end{aligned} \tag{6}$$

This leads to the following state and measurement equations for the covariance $\Sigma(t)_{k,k'}$:

$$\begin{cases} \Sigma(t)_{k,k'} = \Sigma(t-1)_{k,k'} + \left(\eta^2 \nabla \mathcal{L}_{\omega_k(t)} \nabla \mathcal{L}_{\omega_{k'}(t)}\right)) + n_\Sigma \\ l_{k,k'}(t) = \Sigma(t)_{k,k'} - \mu_k(t) \mu_{k'}(t) + \tilde{n}_\Sigma \\ l_{k,k'}(t) = \Sigma(t)_{k,k'} - \mu_k(t) \mu_{k'}(t) + \tilde{n}_\Sigma \end{cases} \tag{7}$$

where n_Σ is the state noise and $\tilde{n}_\Sigma$ is the observation noise, realizations of $\mathcal{N}(0, \sigma_\Sigma^2)$ and $\mathcal{N}(0, \tilde{\sigma}_\Sigma^2)$ respectively.

1.3 TRADI training algorithm overview

We detail the TRADI steps during training in Algorithm 1.

For tracking purposes we must store $\mu_k(t)$ and $\sigma_k(t)$ for all the weights of the network. Hence, we are computationally lighter than Deep Ensembles, which has a training complexity scaling with the number of considered models. In addition, TRADI can be applied to any DNN without any modification of the architecture, contrarily to MC dropout that requires adding dropout layers to the underlying DNN. For clarity we define $\mathcal{L}(\,!\,(t), B(t)) = \frac{1}{jB(t)j} \sum_{(x_i,y_i)2\,B(t)} \mathcal{L}(\,!\,(t), y_i)$. Here $\mathbf{P}_\mu$, $\mathbf{P}_\sigma$ are the noise covariance matrices of the mean and variance respectively and $\mathbf{Q}_\mu$, $\mathbf{Q}_\sigma$ are the optimal gain matrices of the mean and variance respectively. These matrices are used during Kalman filtering [4].

1.4 TRADI parameters

We have set the number of random projections[5] $N = 10$ in all experiments in order to get a fast approximation of the covariance matrix. We validated this choice experimentally and noticed that the performance is similar for larger values of N. $N = 10$ ensures a relatively low computational cost. We used $\sigma_{\text{rbf}} = 1$ for the RBF parameter of the random projection. We have tested different values, without substantial changes in the results. As it can be seen in the algorithm section we have performed a weighted average between the estimated variance/mean with the tracked variance/mean, where the weight depends on Kalman gain.

2 Complementary results

In this section, we detail some of the results reported in the main article for the OOD experiments. The major interest of OOD experiments is that they allow one to see how much we can rely on a DNN. This question is also crucial for industrial research. In this scenario, a particular DNN is trained for a specific application/context which takes into account a certain number of classes. However, in the testing phase new unforeseen objects may appear, potentially leading to wrong/dangerous decisions if the DNN confidence is badly calibrated.

2.1 Results on MNIST

Figure 1 shows the calibration plots for the OOD experiments with MNIST and NotMNIST datasets. As one can see, our strategy (blue curve) has better performances on predicting OOD classes. Calibration plots can easily show whether a DNN is overconfident or not and give an idea on how reliable are the predictions of the DNN. From these plots we see that Deep Ensembles and MC dropout are overconfident, hence they classify non-digits with wrong classes and with high confidence. Our strategy is therefore more suitable for this problem, although still improvable in the lower confidence ranges.

Algorithm 1 TRADI algorithm during training

$\boldsymbol{\omega}(t)$: weights, η learning rate, σ_μ, $\tilde{\sigma}_\mu$, σ_σ, $\tilde{\sigma}_\sigma$
$\mathbf{P}_\mu(0) = \mathbf{0}$, $\mathbf{P}_\sigma(0) = \mathbf{0}$, $\boldsymbol{\omega}(0)$, $t = 1$
for $B(t)$ *2* data **to do**
    begin
    (Forward pass)
    *8*x_i *2* $B(t)$ calculate $g_{\boldsymbol{\omega}(t)}(x_i)$
    evaluate the loss *L*$(\boldsymbol{\omega}(t), B(t))$
    (Backward)
    for $k = 1$ **to** K **do**
        begin
        $\omega_k(t)$ $\omega_k(t$ $1)$ η *r L*$_{\omega_k(t)}$
        end
    (Tracking with Kalman filter)
    for $k = 1$ **to** K **do**
        begin
        # Update predicted (a priori) estimate covariances
        $\mathbf{P}_\mu(t^-)$ $\mathbf{P}_\mu(t$ $1) + \sigma_\mu$
        $\mathbf{P}_\sigma(t^-)$ $\mathbf{P}_\sigma(t$ $1) + \sigma_\sigma$
        # Update Kalman Gains
        $\mathbf{Q}_\mu$ $\mathbf{P}_\mu(t^-)/(\mathbf{P}_\mu(t^-) + \tilde{\sigma}_\mu)$
        $\mathbf{Q}_\sigma$ $\mathbf{P}_\sigma(t^-)/(\mathbf{P}_\sigma(t^-) + \tilde{\sigma}_\sigma)$
        # Update mean
        $\mu_k(t^-)$ $\mu_k(t$ $1)$ η *r L*$_{\omega_k(t)}$
        $\mu_k(t)$ $(1$ $\mathbf{Q}_\mu)\mu_k(t^-) + \mathbf{Q}_\mu\omega_k(t)$
        # Update variance
        $\sigma_k^2(t^-)$ $\sigma_k^2(t$ $1) + \eta^2$ (*r L*$_{\omega_k(t)}$ $\mu_k(t))^2$
        $\sigma_k^2(t)$ $(1$ $\mathbf{Q}_\sigma)\sigma_k^2(t^-) + \mathbf{Q}_\sigma(\omega_k(t)^2$ $\mu_k(t)^2)$
        # Update (a posteriori) estimate covariances
        $\mathbf{P}_\mu(t)$ $(1$ $\mathbf{Q}_\mu)\mathbf{P}_\mu(t^-)$
        $\mathbf{P}_\sigma(t)$ $(1$ $\mathbf{Q}_\sigma)P_\sigma(t^-)$
        end
    (Time update)
    t $t + 1$
    end

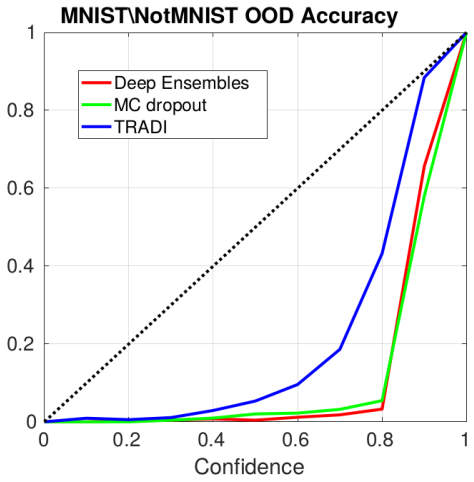

Fig. 1. Calibration plot for MNIST *n*NotMNIST.

	MC dropout	Deep Ensembles	TRADI
mean IoU	0.4857	**0.5719**	0.5298
Accuracy	0.8034	**0.8806**	0.8488

Table 1. CamVid semantic segmentation results (mIoU, accuracy).

2.2 Results on CamVid

We provide additional scores for CamVid experiments. In Figure 2b we illustrate the average precision calibration curve. This curve is similar to the calibration plot, with the difference that for each confidence bin, we do not plot the accuracy but the average precision. The usefulness of the precision is that it highlights more the false-positive effects than the accuracy. We observe in in Figure 2 that TRADI is better on both measures at identifying OOD classes.

In Table 1 we report the mIoU and the global accuracy scores. On these metrics, TRADI is between Deep Ensembles and MC Dropout. In contrast to Deep Ensembles we do not need to train multiple DNNs. In order to achieve good performances on semantic segmentation for complex urban scenes, high capacity DNNs are necessary. Training multiple instances of such networks as in Deep Ensembles brings a significant computational cost. Furthermore, these models are usually updated when new data is recorded and each time the full ensemble needs updating. TRADI requires training a single network each time.

In Figures **??**, **??**, 6, and 7 we report additional qualitative results. Figures 6 and 7 show zoom-in over areas of interest in Figures 4 and 7 respectively. We provide the color code for the semantic segmentation map in Figure 3. We remind that in this experiments the classes human, bicyclist, and car are used as OOD and removed from the train set. We can see that TRADI outputs less confident predictions for human pixels, comparing to Deep Ensebles and MC Dropout.

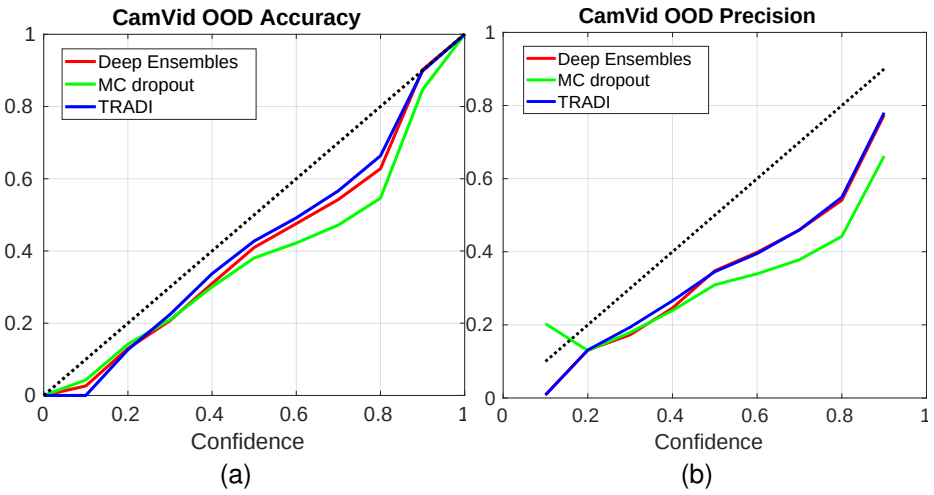

Fig. 2. (a) Calibration plot for the CamVid experiment. (b) Calibration plot, where on the Y axis we replace the Accuracy by the average precision of each class for the CamVid experiment.

Comparing with Deep Ensembles. Deep Ensembles is among the most powerful and effective techniques for epistemic uncertainty. However few works on uncertainty estimation with DNNs on computer vision tasks have considered it for evaluation. We argue that this work is one of the first to truly challenge Deep Ensembles. While we do not achieve higher accuracy than Deep Ensembles, our approach strikes a good compromise between computational cost for training and prediction performance. The computational budget for Deep Ensembles is proportional to the number of models in the ensemble, while for TRADI we always train a single model regardless of the number of network samples we have at test time. Our results on the OOD experiments challenge and sometimes outperform the Deep Ensembles ones.

Fig. 3. Color map for the CamVid experiment.

Fig. 4. Qualitative results on CamVid experiments. Column (1): *top* - input image (the image contrast has been enhanced for clarity with respect to the original dataset image), *bottom* - ground truth; Columns (2-4): *top* - confidence scores from MC dropout, Deep Ensembles and TRADI respectively, *bottom* - corresponding segmentation predictions.

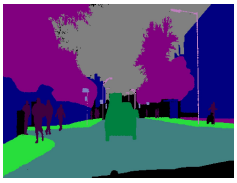

Fig. 5. Qualitative results on CamVid experiments. Column (1): *top* - input image (the image contrast has been enhanced for clarity with respect to the original dataset image), *bottom* - ground truth; Columns (2-4): *top* - confidence scores from MC dropout, Deep Ensembles and TRADI respectively, *bottom* - corresponding segmentation predictions.

Fig. 6. Detailed qualitative analysis of the confidence on CamVid predictions. Column (1): top - input image, bottom - ground truth; Columns (2-4): top - confidence scores from MC dropout, Deep Ensembles and TRADI respectively, bottom - corresponding segmentation predictions.

Fig. 7. Detailed qualitative analysis of the confidence on CamVid predictions. Column (1): top - input image, bottom - ground truth; Columns (2-4): top - confidence scores from MC dropout, Deep Ensembles and TRADI respectively, bottom - corresponding segmentation predictions.

Bibliography

[1] Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M.W., Pfau, D., Schaul, T., Shillingford, B., De Freitas, N.: Learning to learn by gradient descent by gradient descent. In: Advances in neural information processing systems. pp. 3981–3989 (2016)

[2] He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)

[3] Jazwinski, A.H.: Stochastic processes and filtering theory. Courier Corporation (2007)

[4] Kalman, R.E.: A new approach to linear filtering and prediction problems. Journal of basic Engineering **82**(1), 35–45 (1960)

[5] Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: Advances in neural information processing systems. pp. 1177–1184 (2007)

[6] Yang, G.: Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. arXiv preprint arXiv:1902.04760 (2019)