# Review
# Uncertainty Estimation in Deep Networks

***Mohammed El mendili*** *November 2, 2020*

Recent advances in Deep Learning have led to impressive performances on tasks such as object detection, speech recognition, image classification, etc. However, deploying these models in real world applications requires far more than a good test performance. Questions such as explanability, interpretability or uncertainty quickly become crucial when designing an automated decision model. In this review, we will discuss a couple of methods used to **quantify** the notion of **uncertainty** in Deep Neural Network. Applications include detection of out-of-distribution samples, adversarial attacks, or even the famous exploitation/exploration dilemma in Reinforcement Learning (RL) tasks.

## 1 Deterministic and Probabilistic formulations of NNs

Let's denote $\mathbf{D} = (\mathbf{x_i}, \mathbf{y_i})_i$ the training dataset, $\mathbf{w}$ the NN parameters and $f_{\mathbf{w}}(\mathbf{x_i})$ the prediction of the NN for a given input.
The classical idea is to train the NN to fit D as much as possible (while preventing over-fitting and enabling generalization). Although this training formulation is very intuitive, it conceptually fails to help responding to the question **"How much the model is confident about its prediction?"**. In order to quantify the uncertainty, one can see this problem in a probabilistic way. More precisely, we see the NN as a probabilistic model $\mathbf{P}(\mathbf{y}|\mathbf{x}, \mathbf{w})$ (For classification this is a categorical distribution and for regression, this is a Gaussian distribution). At this points, two different ideas are used:

- **MLE (Maximum of Likelihood Estimation)**: We try to find the parameters that maximizes the likelihood of observations.

$$\mathbf{w}^{MLE} = argmax_w log(\mathbf{D}|\mathbf{w}) \tag{1}$$

- **MAP (Maximum a posteriori)**: We try to find the parameters that maximizes the posterior distribution given observations and a prior distribution.

$$\mathbf{w}^{MAP} = argmax_w log(\mathbf{w}|\mathbf{D}) = argmax_w log(\mathbf{D}|\mathbf{w}) + log(\mathbf{P}(\mathbf{w})) \tag{2}$$

  **Remark:** In MAP, a Gaussian prior on w is equivalent to having, in the deterministic approach, a L2 regularization, and a Laplace priors is equivalent to L1 regularization.

Bayesian NN aims to compute the posterior distribution of weights given observations $\mathbf{P}(\mathbf{w}|\mathbf{D})$ and then provides prediction using a simple expectation under this distribution (equivalent to ensembling an infinity NN models). This is intractable in practice and some methods were designing to alleviate this problem.

## 2 Bayes by backrop [1]

The previous probabilistic formulation arises many computational problems: intractable posteriors, the analytical form of NN $f_{\mathbf{w}}(\mathbf{x})$ isn't easily integrable, etc. In [1], authors suggested a **variational approximation** to this posterior distribution. The idea is to use a *simpler* distribution $q(\mathbf{w}, \theta)$ and try to make it as close as possible to the real one. One way to perform this task is to minimize the **Kullback-Leiber** (KL) divergence:

$$\theta = argmin_\theta KL(q(\mathbf{w}|\theta)||\mathbf{P}(\mathbf{w}|\mathbf{D})) = argmin_\theta \int q(\mathbf{w}|\theta) log(\frac{q(\mathbf{w}|\theta)}{\mathbf{P}(\mathbf{w})\mathbf{P}(\mathbf{D}|\mathbf{w})}) d\mathbf{w} \tag{3}$$

While many works in literature tried to find a somehow closed form to the above function (let's denote it $F(\mathbf{D}, \theta)$), authors in [1] chose to approximate it using a simple Monte-carlo simulator by sampling $(w^1, ..., w^n)$ from q:

$$F(\mathbf{D}, \theta) \approx \sum_{i=1}^{n} log(q(w^i|\theta) - log(\mathbf{P}(w^i)) - log(\mathbf{P}(\mathbf{D}|w^i)) \tag{4}$$

This approximation allows a to use any prior and variational distributions (authors in [1] showed that this doesn't yield to any decrease of performance in their experiments compared to the cases where a closed form is available). Having this approximation, we can perform the optimization using Gradient Descent/Backpropagation as used in classical NN trainings. Note also that this is also compatible with minibatches optimization (as $log(\mathbf{P}(\mathbf{D}|w^i)) = \sum_{(x,y)\in\mathbf{D}} log(\mathbf{P}(\mathbf{y}|x, w^i))$).

## 3 Uncertainty Estimation for Out-of-Distribution (OOD) detection [2,3]

One of the most important applications of uncertainty estimation is the detection of **outliers** input data (i.e. drawn away from the training data statistically or adversarially). As a general rule, Deep NNs with a softmax classifier tend to be overconfident for unseen data which justifies the need of using uncertainty when deploying these models in real-world applications. In this section, we will be presenting two comparable approaches to solve this problems. Note that we will restrict our study on classification problems (C={1,...,K} is the set of classes).

### 3.1 Class conditional Gaussian Distribution based approach [2]

In [2], authors used the following simple, yet fundamental equivalence between LDA (Linear discriminant analysis) and the softmax classifier:

$$\exists\Gamma, \forall k, \exists\mu_k, x|y = k \sim \mathcal{N}(\mu_k, \Gamma) \Leftrightarrow \forall k, \exists\beta_k, b_k, \mathbf{P}(y = k|x) = \frac{exp(\beta_k x + b_k)}{\sum_{c\in C} exp(\beta_c x + b_c)}$$

Given a **well-trained** NN with a softmax classifier (in the last layer), the idea here is to use the features given by the other layers to assume a conditional class Gaussian distribution as defined in LDA. More formally, let's denote $g_{w,i}(x)$ the projection of x into the space generated by the i-th layer of the NN (**already trained** with $w$ as a parameter vector. We assume the following:

$$\forall i, k, g_{w,i}(x)|y = k \sim \mathcal{N}(\mu_{k,i}, \Gamma_i)$$

where $\mu_{k,i}$ and $\Gamma_i$ are respectively set to the empirical means (of $g_{w,i}(x)$) of samples labeled in the given class and to the empirical covariance matrix for the whole (projected) dataset.
Given these assumptions, we define, for each layer of the NN, the **Mahalanobis distance-based confidence score** as follows:

$$\forall i \in L, M_i(x) = max_c - (g_{w,i}(x) - \mu_c)^\top \Gamma_i^{-1}(g_{w,i}(x) - \mu_c)$$

which corresponds to the log of the probability density (given training data) of the test sample considered. The reason why we avoid working directly on the feature given by the last softmax layer is because of the so-called problem *label-overfitting* (high confidence for abnormal samples) that the softmax exhibits.
After computing the scores for each layer, the idea is to perform a logistic regression on these scores (as features) to detect whether a sample is OOD or not. This defines the final score as follows:

$$M(x) = \sum_i a_i M_i$$

where $a_i$ are the learnt parameters of the log reg model. In [2], calibration methods such as input pre-processing (by adding a noise to OOD samples) were performed.

### 3.2 Ensembling Self Supervised Leave-out Classifiers [3]

Ensembling is known to perform well on many Machine Learning tasks as it helps reduce variance and prevents overfitting (e.g. Decision Trees Vs Random Forest). The idea here is to ensemble different classifiers trained on different **parts** of the dataset. More precisely, we divide our dataset $D = [D_1, ..., D_K]$ to K mutual exclusive parts (i.e. no overlap between labels in $D_i$ and $D_j$ for $i \neq j$). For the classifier $i$, $D_i$ classes are labeled also as OOD (denoted $X_{od}$) and the others classes in the other parts of D are considered as In-Distribution (ID) (denoted $X_{in}$). Using this paritition, each classifier $i$ learns to detect OOD samples by minimizing the following margin entropy loss ($\beta$ and $m$ are hyper-parameters)

$$L_i = \underbrace{-\frac{1}{|X_{in}|}\sum_{x\in X_{in}} log(f_w(x))}_{\textbf{cross entropy term}} + \overbrace{\beta max(m + \frac{\sum_{x\in X_{in}} f_w(x)log(f_w(x))}{|X_{in}|} - \frac{\sum_{x\in X_{ood}} f_w(x)log(f_w(x))}{|X_{ood}|}, 0)}^{\textbf{Margin of at least m between entropies for ID and OOD samples}} \tag{5}$$

The entropy in the second term encourages the probabilities of the softmax to non-ground truth to decrease while the cross-entropy in the first term increases the probability for ground-truth classes.
This leads, by ensembling, to define our classifier on our dataset. Given this classifier, we can compute the OOD detection score as the average over classifiers of the sum of the maximum probability of the softmax layer and to negative entropy and then define a threshold for classifying OOD samples.

## 4 Results and Remarks

We derive the following remarks and results:

- In [2], authors trained their model on some datasets (for e.g. CIFAR-100) and tried to detect adversarial attacks from **independent** datasets (BIM, DeepFool, etc) . Their results outperformed many state-of-the-art models when using the same training procedure. The same remark holds for [3] where this model outperformed the ODIN [4] model for different model architectures and different pairs of dataset (ID, OOD).

- *Bayes by backrop* and *Ensembling Self Supervised Leave-out Classifiers* are similar in the sense that both made a modification on the loss function to incorporate the notion of uncertainty. In contrast, the *Class conditional Gaussian Distribution based approach* works directly on an already well-trained NN. This presents a big computational gain as we don't have to retrain the NN to extract the uncertainties.

- *Bayes by backrop* seems to be a good alternative to classical NN. However, training a NN with Dropout (on all layers) gives a comparable performances in half time. A more detailed analysis should be made to justify the superiority of this approach.

## References

[1] Blundell et al., *"Weight uncertainty in neural network. ICML 2015.*

[2] Lee et al., *"A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks", NeurIPS 2018*

[3] Vias et al., *"Out-of-Distribution Detection Using an Ensemble of Self Supervised Leave-out Classifiers", ECCV 2018*

[4] Liang et al., *Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks, ICLR 2018*