Bayesian Models
Bayesian Neural Networks
Bayesian Deep Learning and Robustness

Auteur
Youva Addad

Enseignant
Nicolas Thome
Remy Sun
Charles Corbiere

Table des matières

1 Bayesian Models

From posterior $p(w/X, Y) \Rightarrow$ compute predictive distribution given new input x^* : $p\left(\mathrm{y}^*/\mathrm{x}^*, \mathrm{Y}, \mathrm{X}\right) = \int p\left(\mathrm{y}^*, \mathrm{w}/\mathrm{x}^*, \mathrm{Y}, \mathrm{X}\right) dw$

$$p\left(\mathrm{y}^*/x^*, \mathrm{Y}, \mathrm{X}\right) = \int p\left(\mathrm{y}^*/\mathrm{x}^*, \mathrm{w}\right) p(\mathrm{w}/\mathrm{X}, \mathrm{Y}) dw = E_{p(w|\mathcal{D})}\left[p\left(y^* \mid \mathrm{x}^*, \mathrm{w}\right)\right] \tag{1}$$

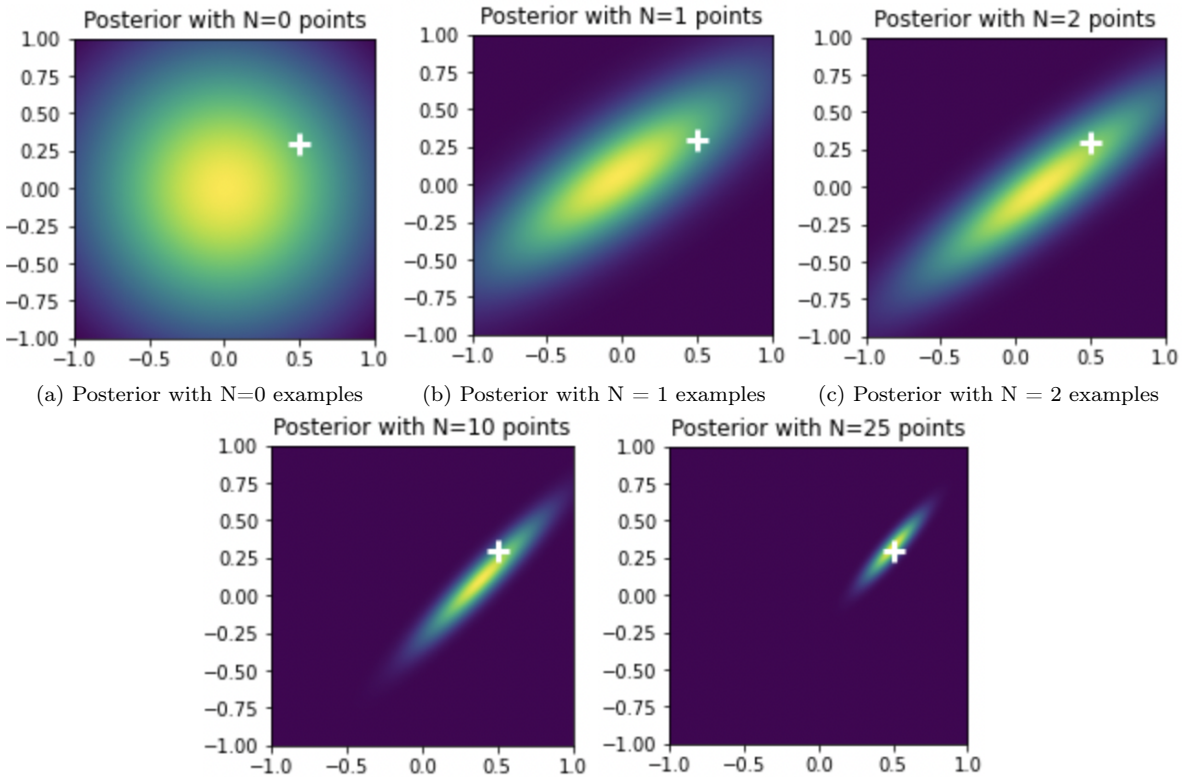
RECAP : for uncertainty estimate with Bayesian models

1. Define prior $p(\mathrm{w})$ and likelihood $p(\mathrm{Y}/\mathrm{X}, \mathrm{w})$
2. Compute posterior distribution $p(\mathrm{w}/\mathrm{X}, \mathrm{Y})$
3. Compute predictive distribution $p\left(\mathrm{y}^*/\mathrm{x}^*, \mathrm{Y}, \mathrm{X}\right)$

1.1 Recall closed form of the posterior distribution in linear case :

$$\begin{aligned} p(\mathrm{w}/\mathrm{X}, \mathrm{Y}) &= \mathcal{N}(\mathrm{w} \mid \boldsymbol{\mu}, \Sigma) \\ \Sigma^{-1} &= \alpha \boldsymbol{I} + \beta \Phi^T \Phi \\ \boldsymbol{\mu} &= \beta \Sigma \Phi^T \mathrm{Y} \end{aligned} \tag{2}$$

Since a Gaussian prior leads to a Gaussian posterior, this means the Gaussian distribution is the conjugate prior for linear regression, β is noise precision and α is variance of parameter w in prior.



(a) Posterior with N=0 examples (b) Posterior with N = 1 examples (c) Posterior with N = 2 examples

(d) Posterior with N = 10 examples (e) Posterior with N = 25 examples

FIGURE 1 – Posterior with different number of examples

1.2 Looking at the visualization of the posterior above, what can you say :

The first figure (N = 0) corresponds to the situation before any data points are observed and shows a plot of the prior distribution in w (posterior reverts to the prior), we can see in figures that more we have data more we approach the white cross (w optimum). With infinite points posterior is a delta function centered at true parameters (white cross)

1.3 Predictive distribution in linear case :

$$p\left(y^* \mid x^*, w, \beta\right) = \mathcal{N}\left(y^*; \Phi\left(x^*\right)^T w, \beta^{-1}\right) : \text{ likelihood} \tag{3}$$

$$p(\mathrm{w} \mid \mathcal{D}, \alpha, \beta) = \mathcal{N}(\mathrm{w}; \mu, \Sigma) : \mathrm{w} \text{ posterior} \tag{4}$$

$$p\left(y^* \mid x^*, \mathcal{D}, \alpha, \beta\right) = \int p\left(y^* \mid x^*, w, \beta\right) p(w \mid \mathcal{D}, \alpha, \beta) dw \tag{5}$$

$$p\left(y^* \mid x^*, \mathcal{D}, \alpha, \beta\right) = \mathcal{N}\left(y^*; \mu^T \Phi\left(x^*\right), \frac{1}{\beta} + \Phi\left(x^*\right)^T \Sigma \Phi\left(x^*\right)\right) \tag{6}$$

$$\sigma^2_{\text{pred}}\ \left(x^*\right) = \tfrac{1}{\beta} + \Phi\left(x^*\right)^T \sum \Phi\left(x^*\right)$$

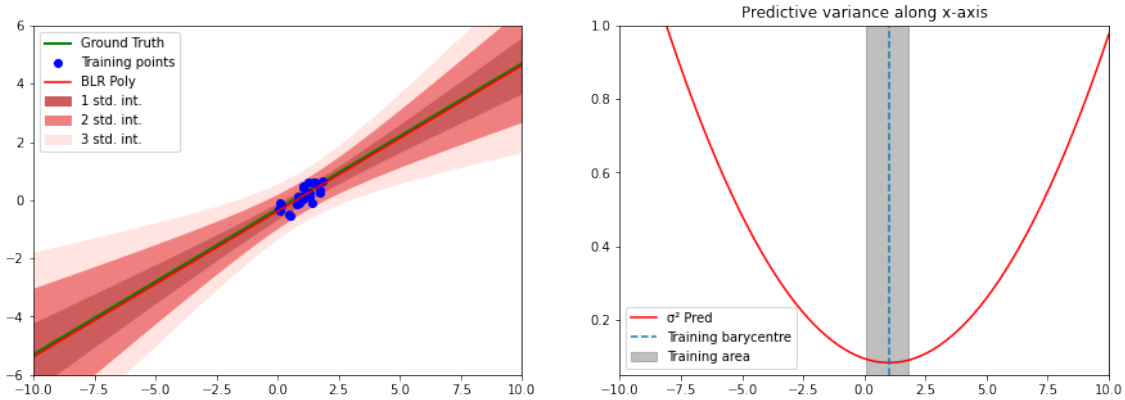


FIGURE 2 – Predictive distribution and uncertainty

1.4 Analyse these results :

In the figure on the left, we visualize the prediction of the model (the red line) and the true line (in green). We notice that the model approximates rather well the true line, but the more one moves away from the barycentre (data center), the more the uncertainty increases (the red surface in left figure). The predicted variance increases when one moves away from the points of the dataset, it reaches its minimum for the barycenter of the data (right figure).

1.4.1 Proof with $\alpha = 0$ and $\beta = 1$:

$$\text{Let } \Phi = \begin{pmatrix} \phi\left(x_1\right)^T \\ \cdots \\ \phi\left(x_N\right)^T \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ \cdots & \cdots \\ 1 & x_N \end{pmatrix}.$$

$$\Sigma^{-1} = \alpha \boldsymbol{I} + \beta \Phi^T \Phi = \begin{pmatrix} \alpha + \beta N & \beta 1^T \mathrm{X} \\ \beta 1^T \mathrm{X} & \alpha + \beta \mathrm{X}^T \mathrm{X} \end{pmatrix} \tag{7}$$

$$\sigma^2_{\text{pred}}\ (\mathrm{x}^*) = \frac{1}{\beta} + \Phi\left(\mathrm{x}^*\right)^T \Sigma \Phi\left(\mathrm{x}^*\right) \tag{8}$$

With with $\alpha = 0$ and $\beta = 1$ we will have $\Sigma^{-1} = \begin{pmatrix} N & 1^T\mathbf{X} \\ 1^T\mathbf{X} & \mathbf{X}^T\mathbf{X} \end{pmatrix}$. Since $X \in R^{N \times 1}$, so $1^T\mathbf{X} = \sum_i^N x_i$ and $\mathbf{X}^T\mathbf{X} = \sum_i^N x_i^2$.

$\Sigma^{-1} = \begin{pmatrix} N & \sum_i^N x_i \\ \sum_i^N x_i & \sum_i^N x_i^2 \end{pmatrix}$.

$\Sigma = \frac{1}{\det(\Sigma^{-1})} \begin{pmatrix} \alpha + \beta X^T X & -\beta 1^T X \\ -\beta 1^T X & \alpha + \beta N \end{pmatrix} = \frac{1}{N \sum_i^N x_i^2 - \left(\sum_i^N x_i\right)^2} \begin{pmatrix} \sum_i^N {x_i}^2 & -\sum_i^N x_i \\ -\sum_i^N x_i & N \end{pmatrix}$, with N nombre of examples, and x_i the example i, with determinant in case of 2D matrix.

$\Phi(x^*)^T \Sigma \Phi(x^*)$ is uncertainty over parameters w let's prove that it increases when x^* far from training data :

$$\Phi(x^*)^T \Sigma \Phi(x^*) = \begin{pmatrix} 1 & x^* \end{pmatrix} \Sigma \begin{pmatrix} 1 \\ x^* \end{pmatrix} = \frac{1}{N\sum_i^N x_i^2 - \left(\sum_i^N x_i\right)^2} (1, x^*) \begin{pmatrix} \sum_i^N {x_i}^2 & -\sum_i^N x_i \\ -\sum_i^N x_i & N \end{pmatrix} \begin{pmatrix} 1 \\ x^* \end{pmatrix}$$

Which give us : $\frac{\sum_i^N x_i^2 - x^* \sum_i^N x_i - x^* \sum_i^N x_i + N(x^*)^2}{N\sum_i^N x_i^2 - \left(\sum_i^N x_i\right)^2} = \frac{\sum_i^N x_i^2 - 2x^* \sum_i^N x_i + N(x^*)^2}{N\sum_i^N x_i^2 - \left(\sum_i^N x_i\right)^2}$

Let show the variance :

$= \frac{N\left(\frac{\sum_i^N {x_i}^2}{N} - \frac{2x^* \sum_i^N x_i}{N} + (x^*)^2\right)}{N^2\left(\frac{\sum_i^N {x_i}^2}{N} - \frac{\left(\sum_i^N x_i\right)^2}{N^2}\right)} = \frac{\frac{\sum_i^N {x_i}^2}{N} - 2x^*\bar{x} + (x^*)^2}{N\left(\frac{\sum_i^N {x_i}^2}{N} - \bar{x}^2\right)}$, with $\bar{x}$ which represent exactly the mean of X, $(x^* - \bar{x})^2 = (x^*)^2 - 2x^*\bar{x} + (\bar{x})^2$.

$= \frac{\frac{\sum_i^N x_i^2}{N} - 2x^*\bar{x} + (x^*)^2 + \bar{x}^2 - \bar{x}^2}{N\,\mathrm{Var}[X]} = \frac{\frac{\sum_i^N x_i^2}{N} - \bar{x}^2 + (x^* - \bar{x})^2}{N\,\mathrm{Var}[X]} = \frac{Var[X] + (x^* - \bar{x})^2}{N\,\mathrm{Var}[X]}$.

$\sigma^2 = 1 + \frac{1}{N} + \frac{(x^* - \bar{x})^2}{N\,\mathrm{Var}[X]}$.

The predicted variance is proportional to the distance between x^* and the barycenter of the data ($\bar{x}$) and is inversely proportional to the size of the dataset and the variance of the dataset. So when $x^* = \bar{x}$ the variance reaches it minimum, which is $1 + \frac{1}{N}$. Moreove, since we have square between the test point and the barycentre, this's why the form of the predictive variance which is parabolic.

1.5 Bonus Question : Bayesian Linear Regression on the following dataset hole :

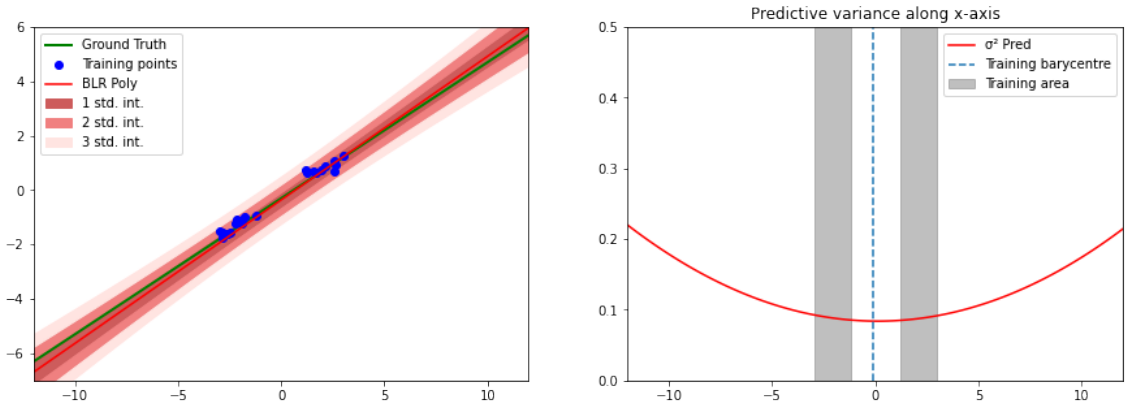


FIGURE 3 – Predictive distribution and uncertainty for hole dataset

The minimum of the predicted variance is still reached for the barycenter of the data, whereas we would like it to be high there because there is no point in this zone. Moreover the variance increases very slowly (less sharp) compare than previous data (right figure). BLR is not adapted to this kind of data, it's insensitive to the position of data here.

1.6 Polynomial basis functions :

$$\Phi = \begin{pmatrix} \phi(x_1)^T \\ \dots \\ \phi(x_n)^T \end{pmatrix} = \begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{D-1} \\ \dots & \dots & \dots & \dots & \\ 1 & x_N & x_N^2 & \dots & x_N^{D-1} \end{pmatrix} \tag{9}$$

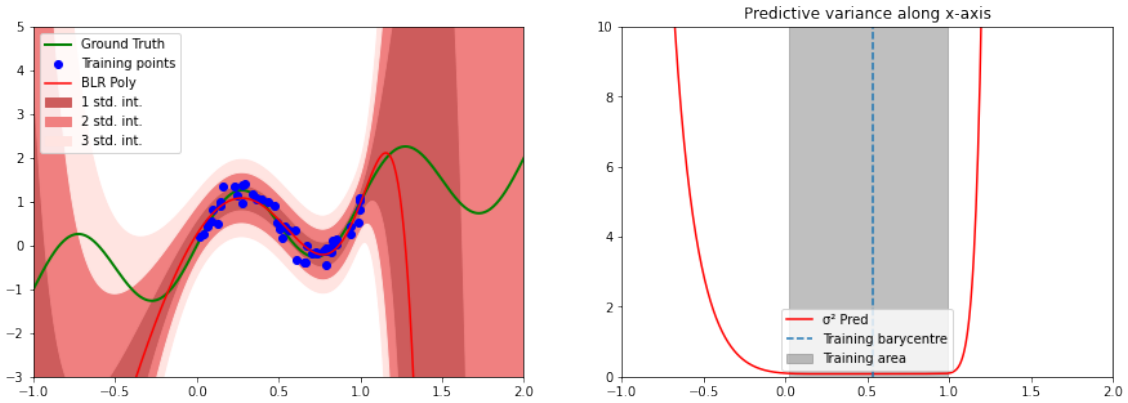


FIGURE 4 – Predictive distribution and uncertainty with polynomial basis functions on sinus dataset

Predictive variance increase as we move away from training data. However here with polynomial features, the minimum is not the training barycentre anymore (we can see the minimum is reached between 0. and 1.). But when we move little away from the data, the predictive variance increases drastically. The red curve (predicted curve) sticks well to the green curve (ground truth) in the presence of the points.

1.7 Gaussian basis functions :

$$\Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_{D-1}(x_1) \\ \dots & \dots & \dots & \dots \\ \phi_0(x_N) & \phi_1(x_N) & \dots & \phi_{D-1}(x_N) \end{pmatrix} = \begin{pmatrix} e^{-\frac{(x_1-\mu_0)^2}{2s^2}} & e^{-\frac{(x_1-\mu_1)^2}{2s^2}} & \dots & e^{-\frac{(x_1-\mu_{D-1})^2}{2s^2}} \\ \dots & \dots & \dots & \dots \\ e^{-\frac{(x_N-\mu_0)^2}{2s^2}} & e^{-\frac{(x_N-\mu_1)^2}{2s^2}} & \dots & e^{-\frac{(x_N-\mu_{D-1})^2}{2s^2}} \end{pmatrix} \tag{10}$$

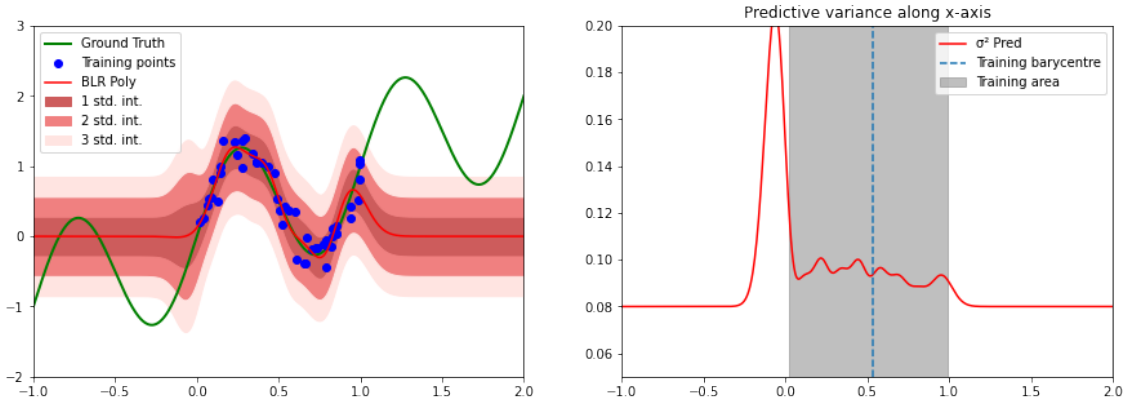


FIGURE 5 – Predictive distribution and uncertainty with gaussian basis functions on sinus dataset

In zone of localization of points, the red curve stick well to the green curve, but this time the predictive variance is not convex (does not reach a global minimum in the localization of points), far from trainig data it tends to $\frac{1}{\beta}$ however, we would like it to tend to infinity.

1.7.1 Why in regions far from training distribution, the predictive variance converges to this particular value :

$\Phi(x^*)^T \sum \Phi(x^*)$ tends to 0 when far from trainig data, so $\sigma^2 = \frac{1}{\beta}$. Since phi is Gaussian basis functions is exponential function, so if we are far from training data $(x^* - \bar{x})^2$ is going to be a very large number, so $exp(-\infty)$ will tends to zéro, $\Phi(x^*)^T \sum \Phi(x^*)$ tends to zéro, so the predictive variance converges to this particular value $\frac{1}{\beta}$.

2 Approximate Inference in Classification

2.1 Maximum-A-Posteriori Estimate :

$$p\left(\mathbf{y}=1 \mid \boldsymbol{x}^{*}, \mathcal{D}\right)=\int p(\mathbf{y}=1 \mid \boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{w} \mid \mathcal{D}) d \boldsymbol{w} \approx p\left(y=1 \mid \boldsymbol{x}, \boldsymbol{w}_{\mathrm{MAP}}\right)$$

$$\boldsymbol{w}_{\mathrm{MAP}}=\arg \max_{\boldsymbol{w}} p(\boldsymbol{w} \mid \mathcal{D})=\arg \max_{\boldsymbol{w}} p(\mathcal{D} \mid \boldsymbol{w}) p(\boldsymbol{w})=\arg \max_{\boldsymbol{w}} \prod_{n=1}^{N} p\left(y_{n} \mid \boldsymbol{x}_{n}, \boldsymbol{w}\right) p(\boldsymbol{w})$$

$$w_{\mathrm{MAP}}=\arg \min_{\boldsymbol{w}} \sum_{n=1}^{N}\left(-y_{n} \log \sigma\left(\boldsymbol{w}^{T} \boldsymbol{x}_{n}+b\right)-\left(1-y_{n}\right) \log \left(1-\sigma\left(\boldsymbol{w}^{T} \boldsymbol{x}_{n}+b\right)\right)+\frac{1}{2 \Sigma_{0}^{2}}\|\boldsymbol{w}\|_{2}^{2}\right)$$

2.1.1 Q1.1-Analyze the results :

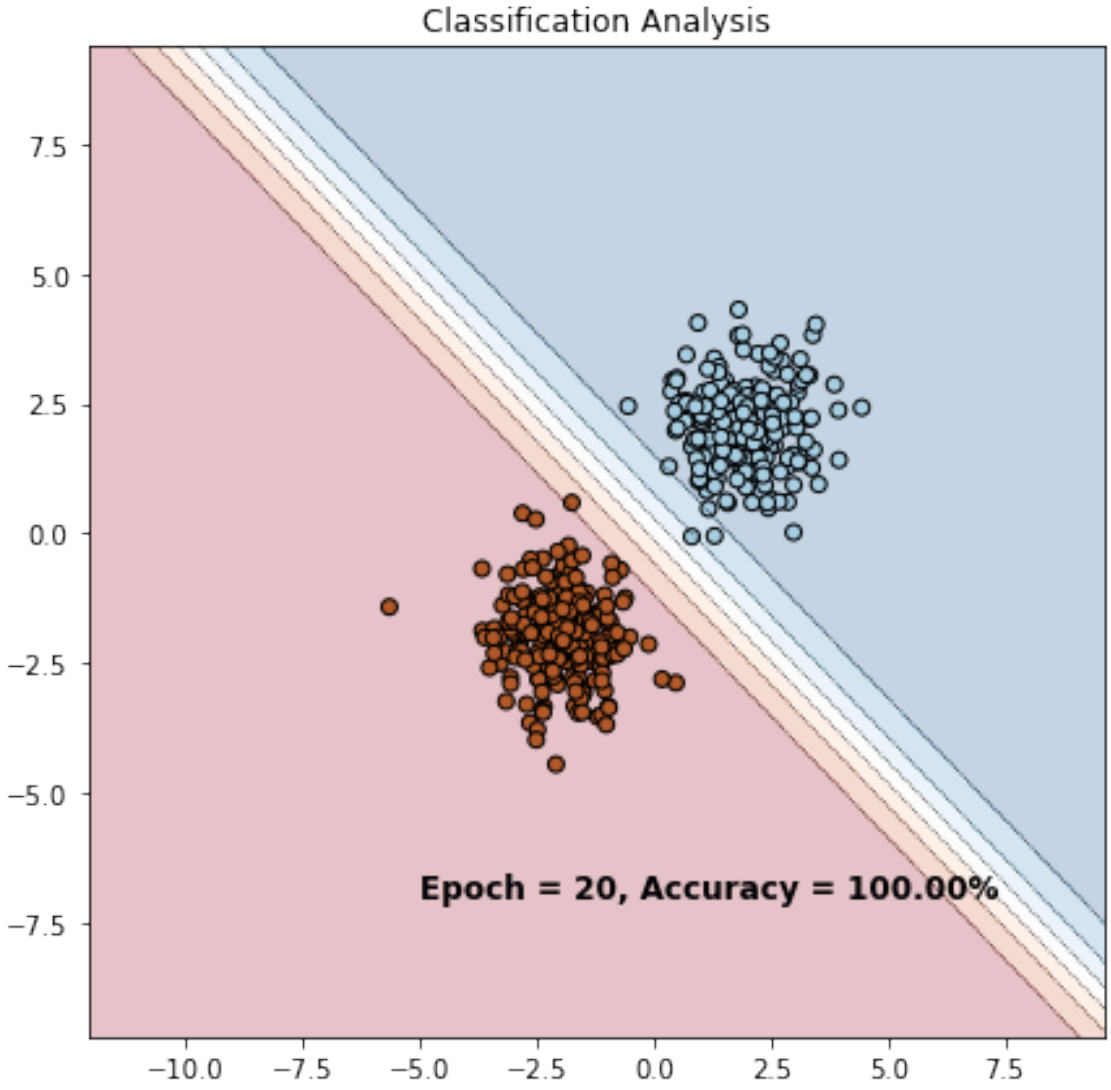


FIGURE 6 – Linear boundary Analysis

First of all, knowing that the data are linearly separable, we notice that the decision boundary divides the data well. However uncertainty does not increase far from training data, the model remains confident but the approximation is not accurate, $p\left(y=1 \mid x, w_{MAP}\right)$ (the right y predicted) is equal to 1 all around even for very far points, except for a very restricted area along the classification boundary.

2.2 Laplace Approximation :

The Laplace approximation aims to find a Gaussian approximation to a continuous distribution. We will use Laplace approximation to estimate the intractable posterior $p(\boldsymbol{w} \mid \mathcal{D})$.

2.2.1 Analyze the results :

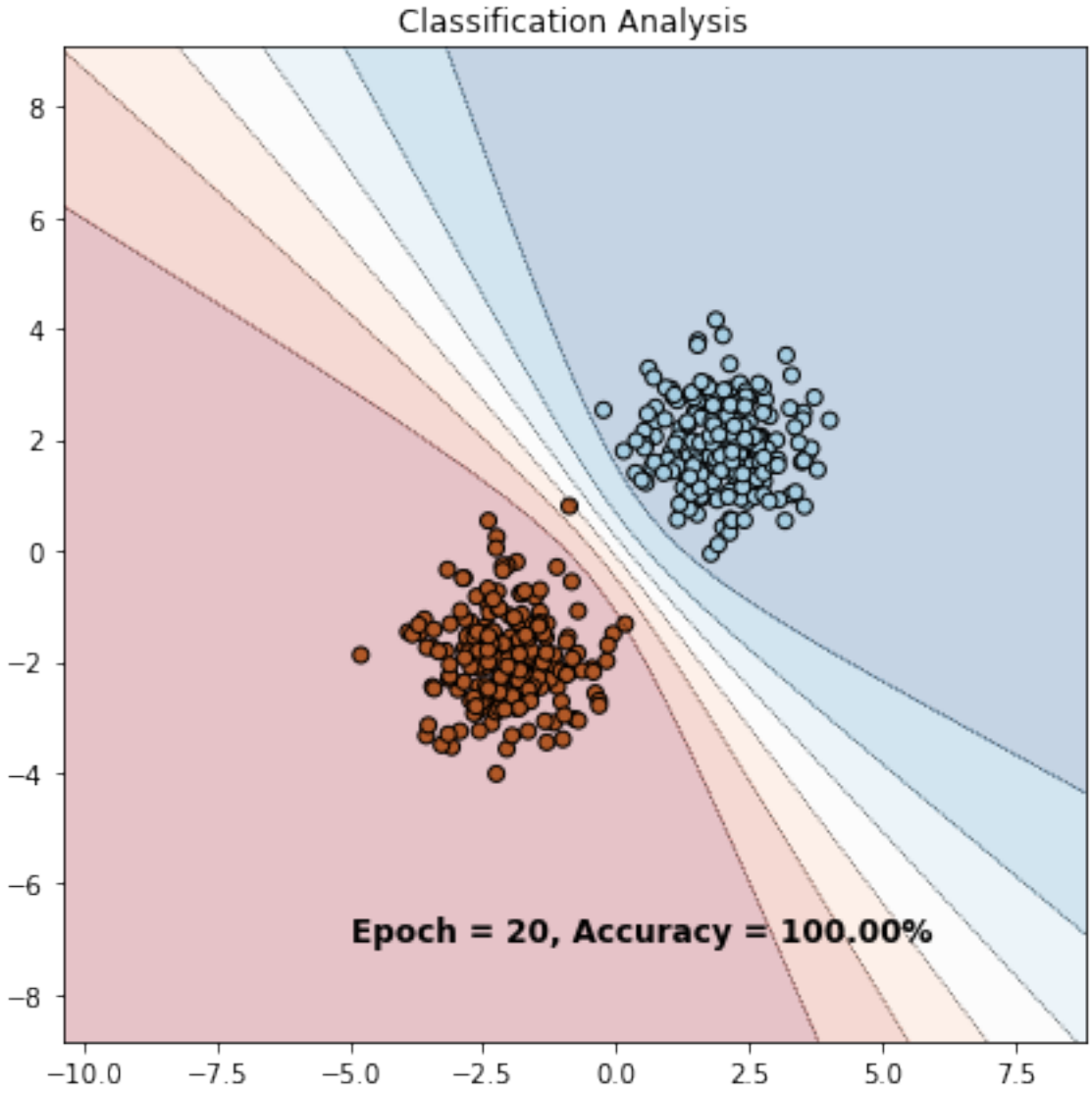


FIGURE 7 – Bayesian Logistic Regression Laplace Approximation

Accuracy still always 100% but now we have uncertainty far from training data, the decision boundary certainly remains linear, but the uncertain is to bend according to this distance to the training points and thus highlights an epistemic uncertainty.

2.3 Variational Inference :

Approximate intractable distribution $p(w \mid D)$ with simpler, tractable distribution $q(w)$. The goal of variational inference is to maximize the variational lower-bound, approximate q distribution, or minimize $\mathrm{KL}(q\|p)$.

$$\mathcal{L}_{VI}(\boldsymbol{\theta}) = \log p(\mathrm{Y} \mid \mathrm{X}) - KL\left(q_\theta(\mathrm{w}) \| p(\mathrm{w} \mid \mathrm{X}, \mathrm{Y})\right) \leq \log p(\mathrm{Y} \mid \mathrm{X}) \tag{11}$$

2.3.1 Analyze the results :

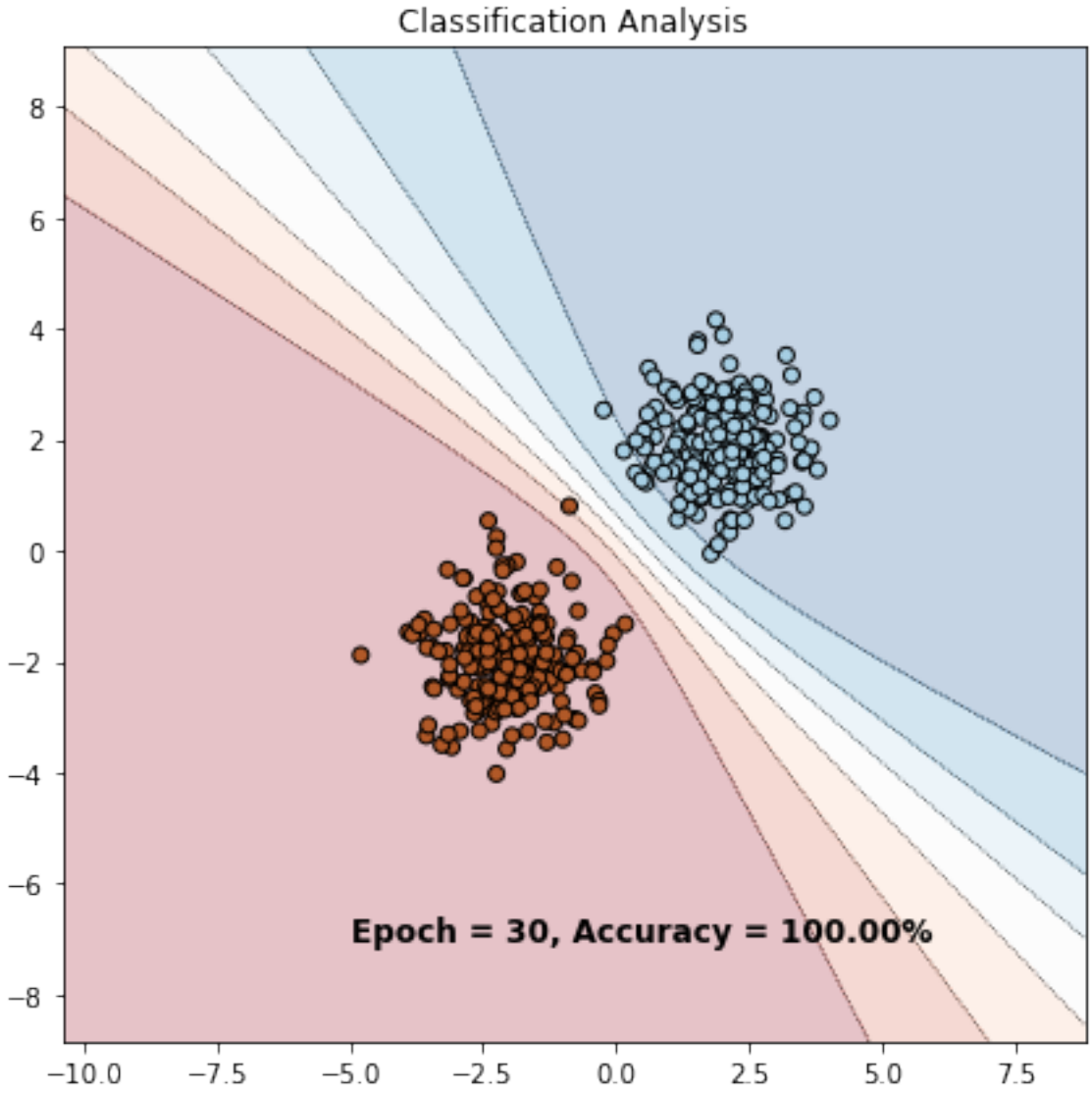


FIGURE 8 – Bayesian Logistic Regression Variational Inference

We can do the same remark as previous, the decision boundary is roughly midway between the groups of data points, and the contours of the predictive distribution deviate from the data reflecting the greater uncertainty in the classification of these regions. The uncertainty of examples away from training points is more pronounced than before in the sense that even examples close to the decision boundary have an uncertainty.
We can clearly see that epistemic uncertainty is much higher in regions of no training data than it is in regions of existing training data.

Comment the class LinearVariational :

- **Parent Attribut** : Allow us to accumulate the KL-divergence of the other LinearVariational modules which are calculated in the case where there are several LinearVariational layers.
- **w_mu, w_rhu, b_mu, b_rhu** : Which represent the mean and variance of weights, to do reparametrization trick to sample from normal which is done ine method sampling.
- **Sampling method** : Use the previous weights to sample from Gaussian distribution, $w_j = \mu_j + \sigma_j \epsilon_j$, the gradient can not flow through the model otherwise. Variance can not be negative we use $\mathbf{\Sigma} = \log(1 + e^{\rho})$.
- **kl_divergence method** : From the inputs w_mu and w_rhu and z provided as input compute $KL(q_\theta(\mathrm{w}) \| p(\mathrm{w}))$, from prior compute from isotropic gaussian, and the normal distribution with w_mu as mean and w_rhu as std.
- **forward** : Sample weight and sample bias, compute the linear output and accumulate KL for weight and bias.

2.4 Bayesian Neural Networks :

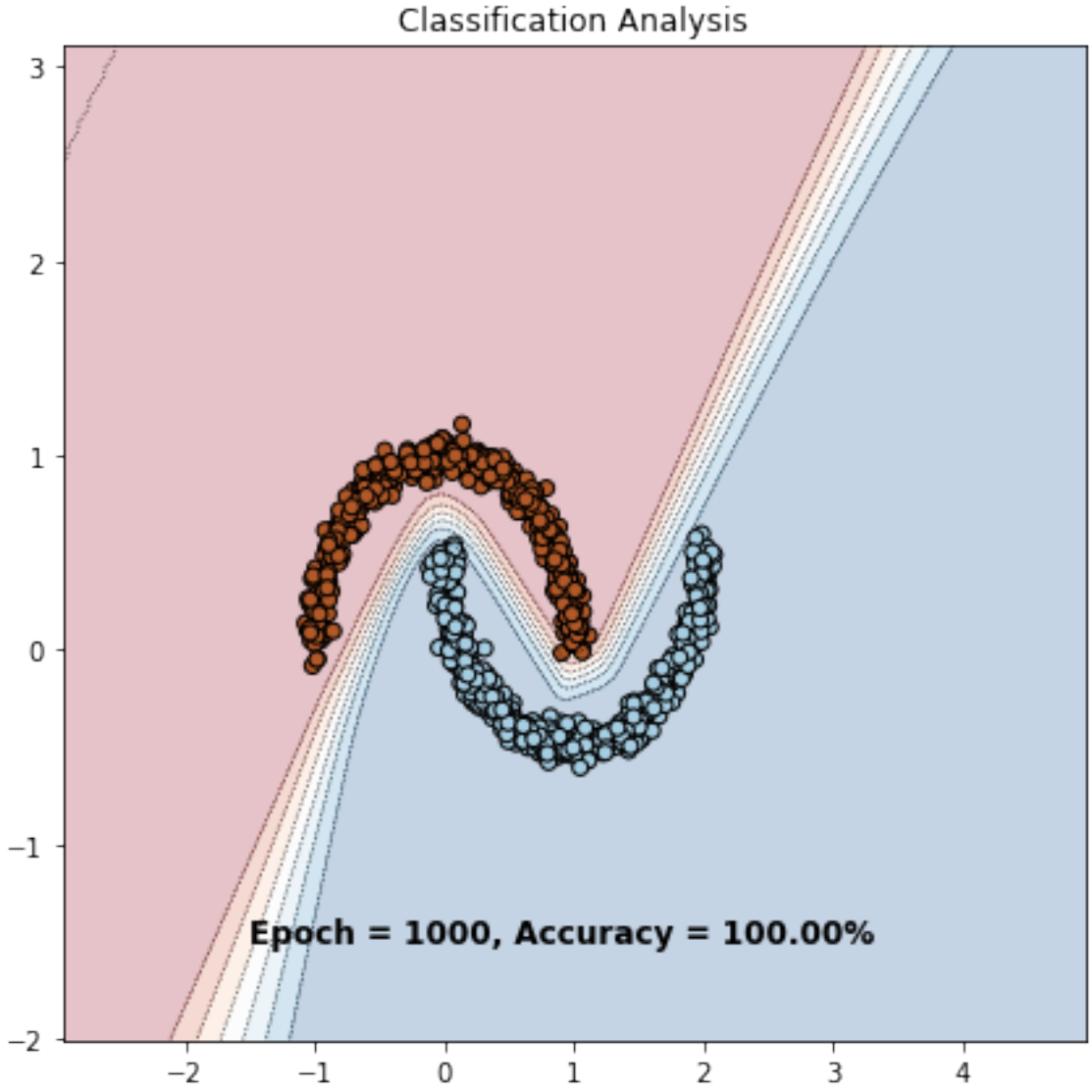


FIGURE 9 – Moons Dataset with Linear Variational

Here we study the moons dataset, we will use LinearVariational layer to define a MLP with 1 hidden layer and ReLU non-linearity. We reach 100% of accuracy with non linear boundary. The same variational Inference approach shows less epistemic uncertainty, coming from a smaller final predictive variance, which clearly happened because of the data types.

2.5 Monte Carlo Dropout :

Training a neural network with randomly dropping some activations, such as with dropout layers, can actually be seen as an approximate variational inference method.

2.5.1 Analyze the results :

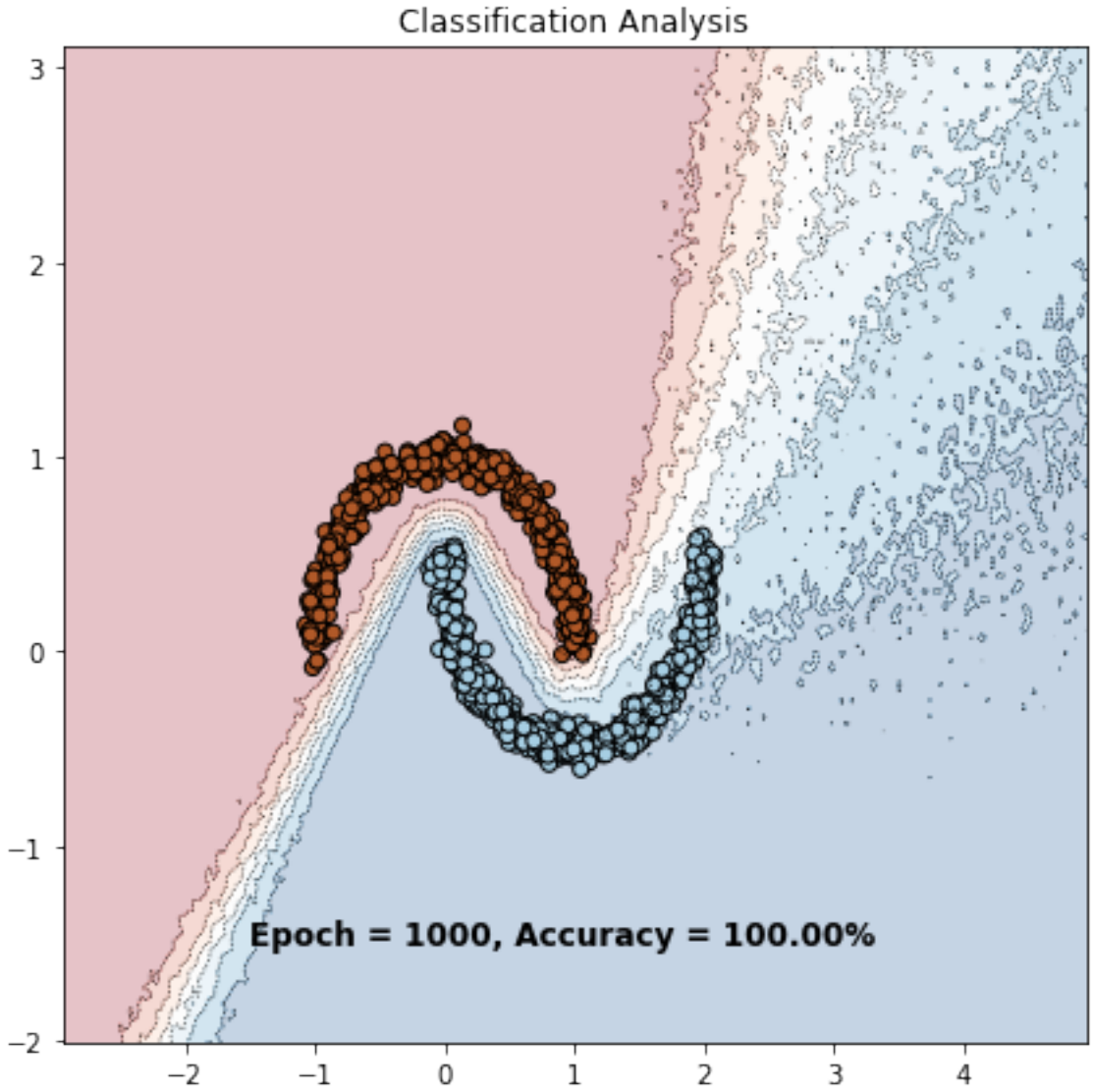


FIGURE 10 – Moons Dataset with MC Dropout

The accuracy still perfect, we can however see an uncertainty more representative to our data, the predictive variance is more conservative and fair with respect to the training data.
Bayesian models usually come with a computational cost. The MC Dropout mitigates the problem of representing uncertainty without sacrificing either computational complexity or test accuracy.

3 Uncertainty Applications

3.1 Monte-Carlo Dropout on MNIST :

3.1.1 Question 1.1 :

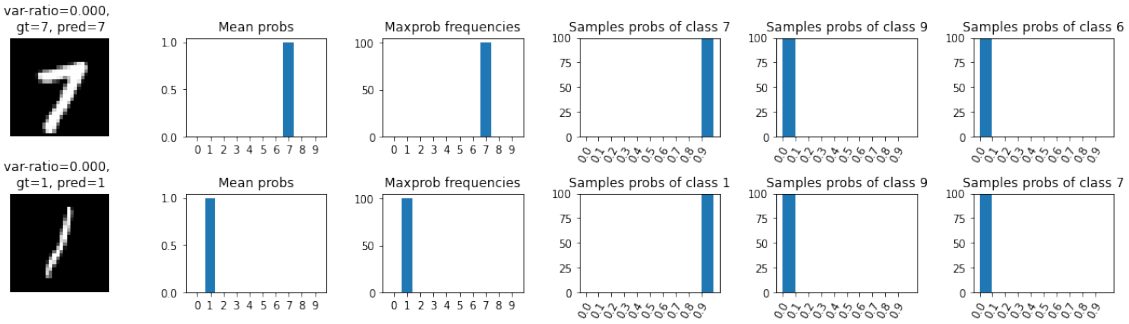


FIGURE 11 – Random uncertain images along with their var-ratios value

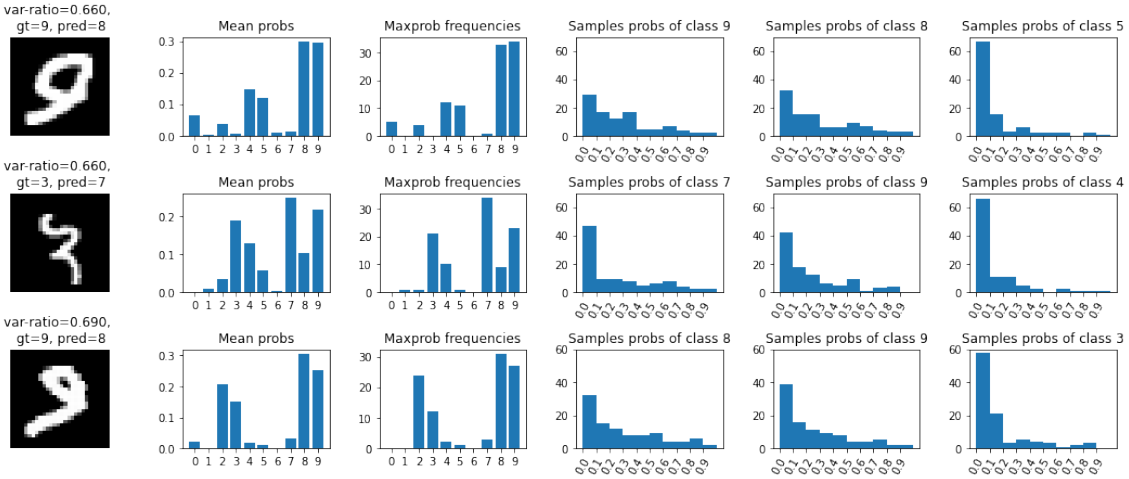


FIGURE 12 – The top 3 most uncertain images along with their var-ratios value

The images themselves are not very recognizable, for the top uncertain images we can see in figure 12 that the confusion can happen on several class (pics in most classes). The model are not confident to which class to classify it, high prediction confidence, low model uncertainty and vice versa.

For the random images there is no ambiguity for the membership to class, and are confident to which class to classify it.

Mean probs represents the mean probability vector and max probability represents how many times a class has been predicted

3.2 Failure prediction :

Our goal is to use uncertainty estimate to accept or reject predictions (distinguish correct from erroneous predictions), Model confidence $\hat{C}(\boldsymbol{x})$:
- Simple baseline for deep neural networks : $\text{MCP}(x) = \max_{k \in \mathcal{Y}} p(Y = k \mid w, x)$
- More advanced methods, e.g. MC dropout for classification.

Rather tha MCP which is overconfident prediction values, rather than the MCP, for ranking confidence measure, it would be better to look at the probability of the class that should have been predicted, True Class Probability (TCP) : $\text{TCP}\,(\boldsymbol{x}, y^*) = p\,(Y = y^* \mid \boldsymbol{w}, \boldsymbol{x})$ unreliable confidence criterion. However TCP is unknown at test time, we'll use ConfidNet, for learning TCP model confidence, which given train set learn a confidence model, its scalar output $\hat{c}(\boldsymbol{x}, \theta)$ close to $TCP\,(x, y^*)$, with TCP more than 0.5 we are sure that the classifier

has the good prediction and on opposite the model will reject prediction.

The objective of this detection is that the model uses this measure of uncertainty to know when it does not know.

3.3 Question 2.1 :

3.3.1 Why did we use AUPR metric instead of standard AUROC :

AUROC is the area under the curve where axis x is false positive rate (FPR) and axis y is true positive rate (TPR), so AUROC is used if our dataset is quite balanced. But in failure prediction, the misclassification are used as the positive detection of the class (error detection), so here we care more about the positive class (highly imbalanced dataset since we have less error with initial classification MNIST), this's why using AUPR, which is more sensitive to the improvements for the positive class.

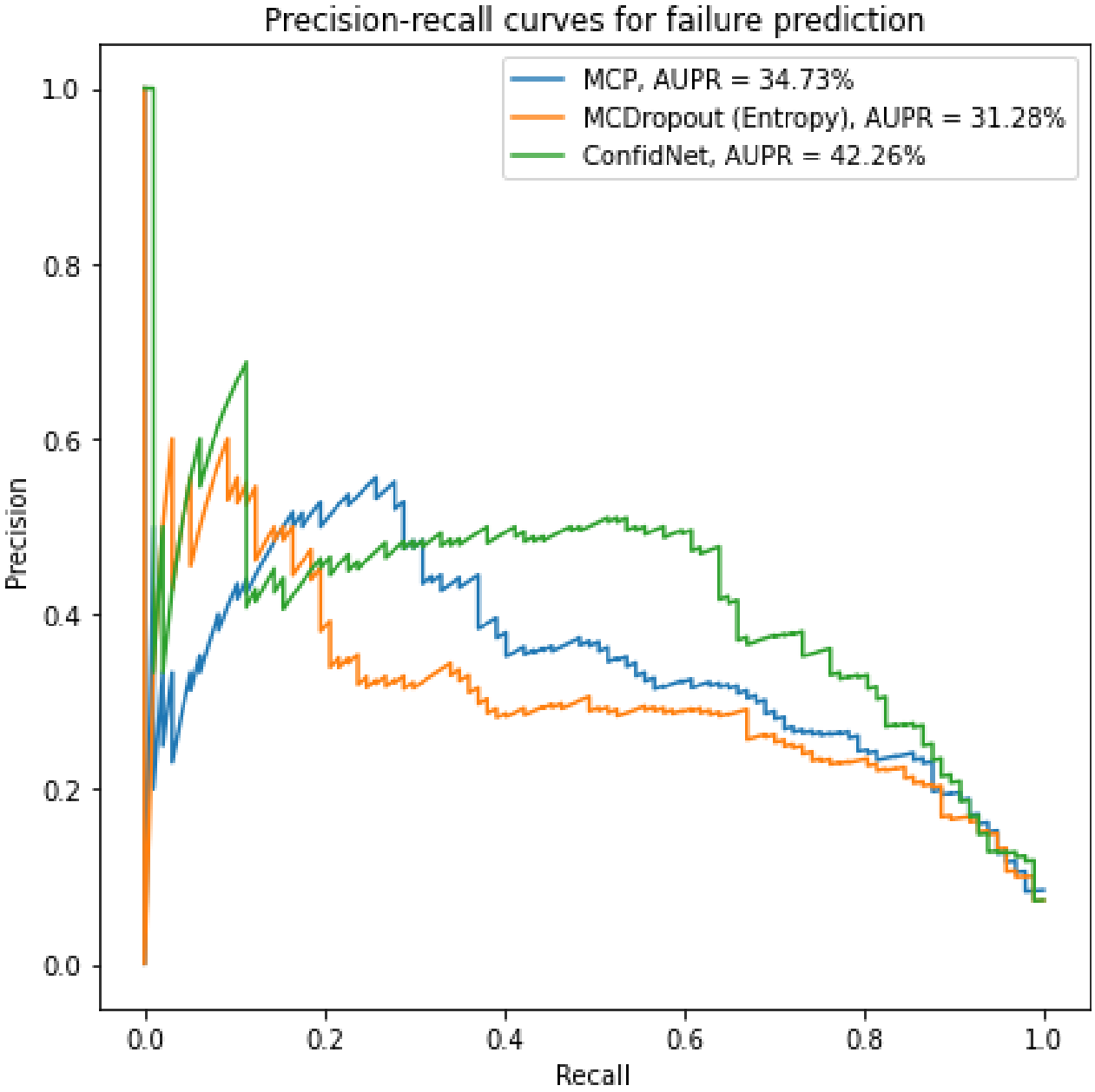


FIGURE 13 – Precion-recall curves for failure prediction

We can remark in figure 13, that the confidNet is the best compare to the other models, for detecting 40% of errors confidNet is $\sim$ 50% precise, it has better precision when the required recall is low, but for detecting 100% of errors all models converges to the same value which is $\sim$ 10%.

We can observe also that for the three methods used, 34.74% for the MCP, 31.28% for the MCDropout entropy and 42.26% for confidnet which does not represent half of the error detection.

3.4 Out-of-distribution detection :

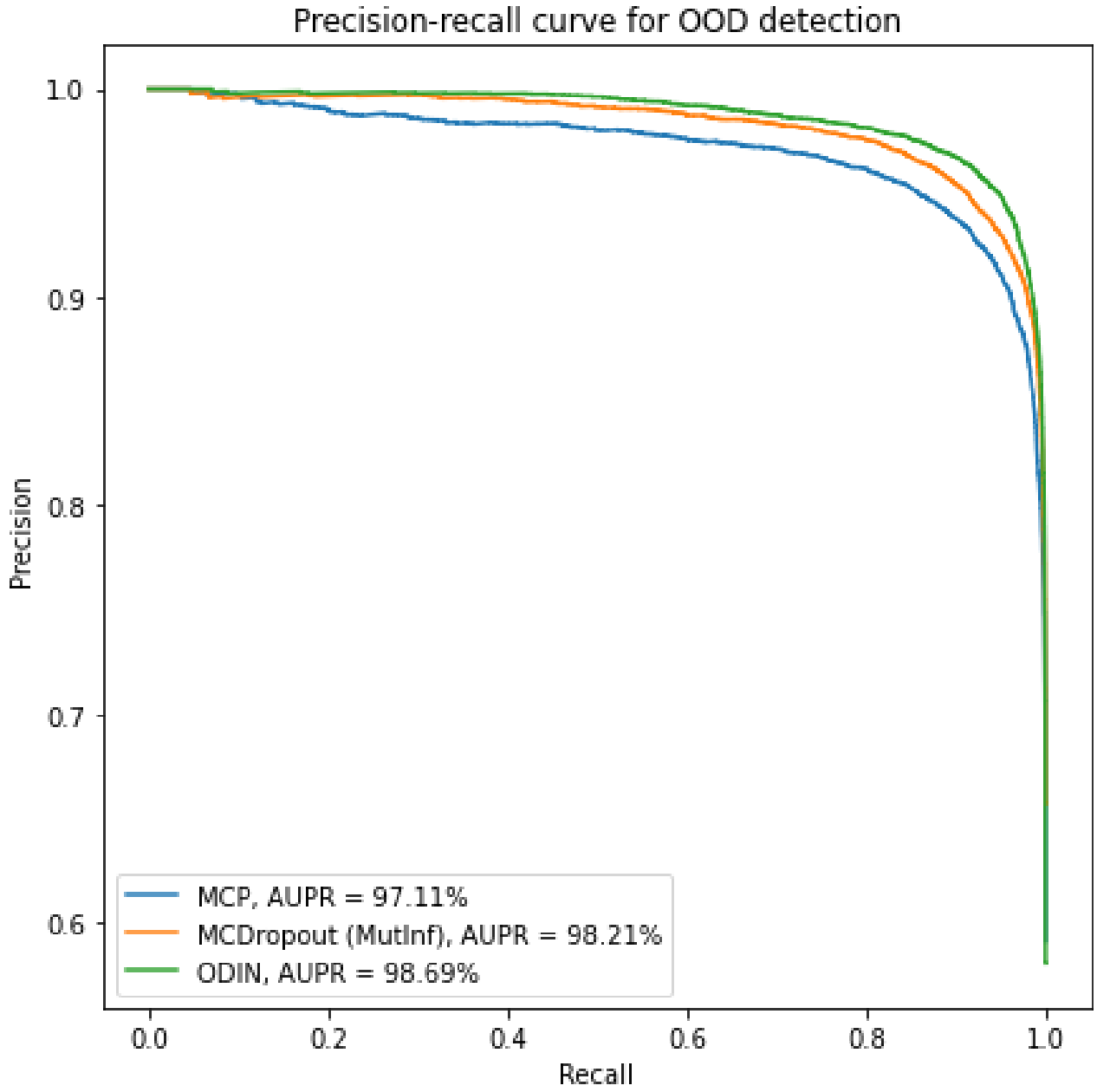


FIGURE 14 – Precion-recall curves for Out-of-distribution detection

We notice in the precision/recall curves that all networks are efficient, we have more than 97% of detection, the curves stretch almost perfectly towards the top-right side of the graph. For 90% of recall we have $\sim$ 95% for MCP, $\sim$ 97% for MCDroupout and $\sim$ 100% for ODIN. we can also see that ODIN has the best performance for Out-of-distribution detection.