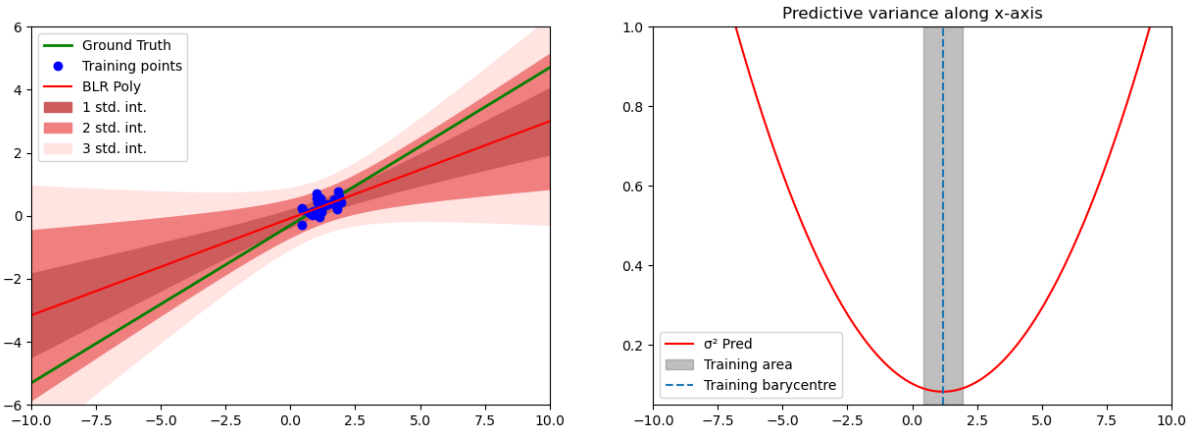
Robust Deep Learning – report

Student: David Faget Caño *Teacher:* Nicolas Thome

❖ Week 1 – Bayesian Models and Uncertainty

Answers to questions that do not appear in this report can be found in the .ipynb file.

[Question 1.4] After doing the predictions on the test dataset, we observe the following results:



We observe that, as we expected, predictive variance increases far from training area. Moreover, its minimum is reached right at training barycentre.

[Question 1.5] Let's do the requested theoretical analysis. Suppose that $\alpha = 0$ and $\beta = 1$. Using the notation employed in class, we have that

$$\sigma^2_{pred}(x^*) = 1 + \Phi(x^*)^T \Sigma \Phi(x^*)$$

Moreover, as seen in class,

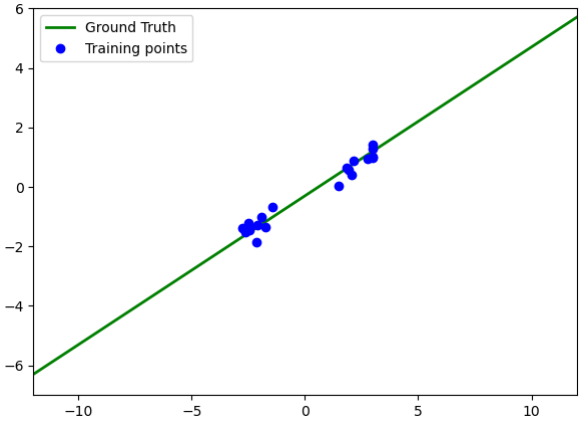
$$\Sigma^{-1} = \begin{pmatrix} N & 1^T X \\ 1^T X & X^T X \end{pmatrix}$$

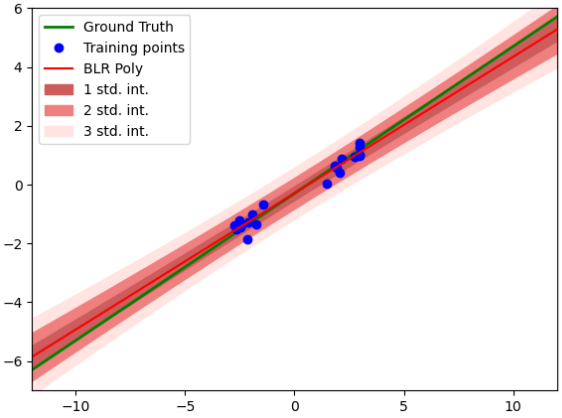
But the inverse of this matrix gives less weight to points which are far from training distribution. As a consequence, $\Phi(x^*)^T \Sigma \Phi(x^*)$ increases when x^* is distant from training data, and $\sigma^2_{pred}(x^*)$ too.

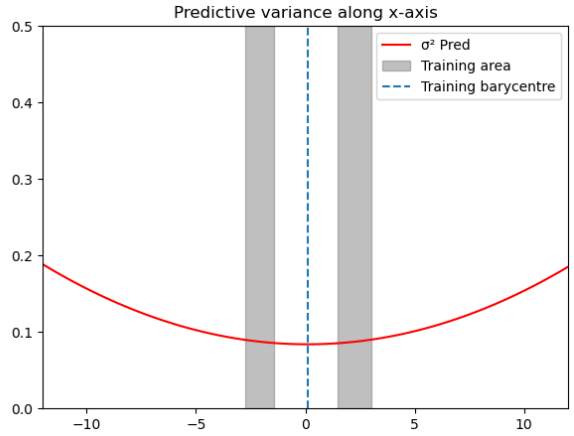
Intuitively, as we move far from the training distribution, the basis functions at the new point become less correlated with the basis functions at the training points. This lack of correlation contributes to an increase in the predictive variance, reflecting the model's uncertainty in

regions where it has not observed much data.

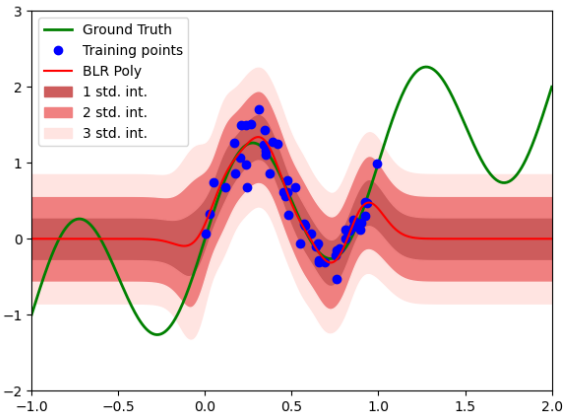
[Bonus Question] When applying Bayesian Linear Regression on the following dataset, it happens that predictive variance still increase far from training area, but much less than for the precedent dataset. This is a consequence of the separation between the two clouds of points. However, its minimum is still reached at training barycentre.

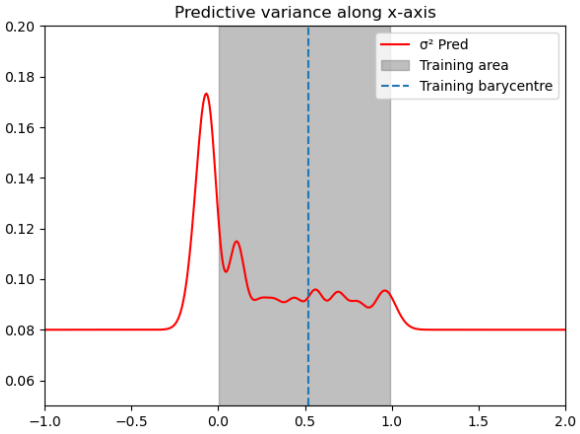






[Question 2.4/2.5] Results obtained on sinusoidal dataset using Gaussian basis functions are the following:





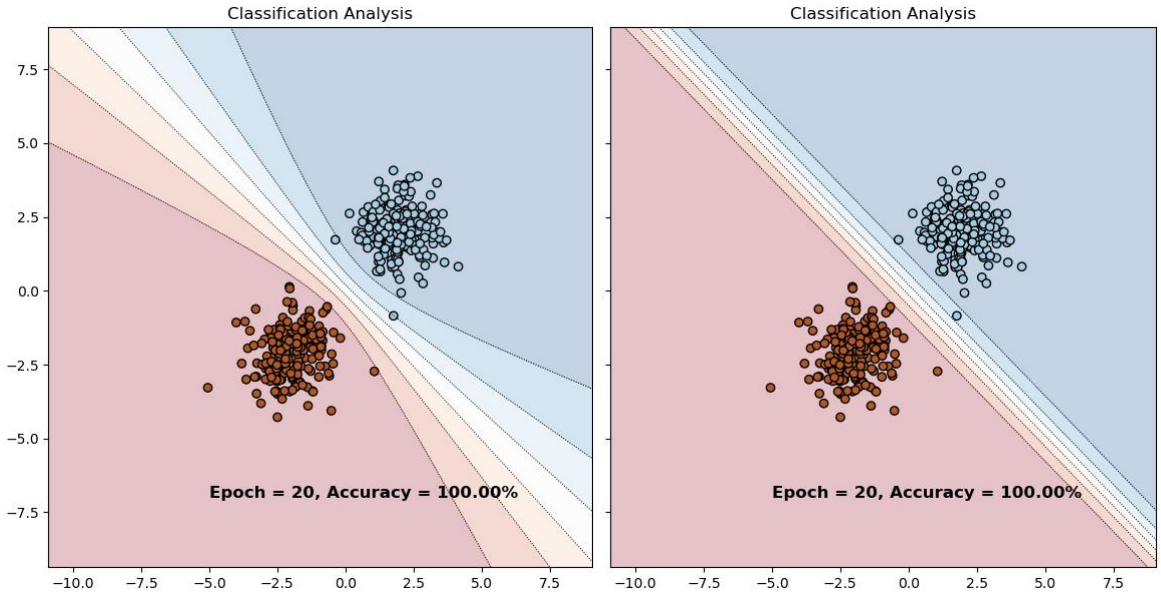
We observe that, this time, predictive variance does not increase far from training distribution. Indeed, it converges to a fixed value (near 0) and it presents higher values near from training distribution. This is because the epistemic uncertainty converges to 0 far from training data. When using localized basis functions like Gaussians, the model's predictions are influenced by the data points within the vicinity of the prediction point. The influence of each data point is weighted by the value of the corresponding basis function at that point. In regions far from the training distribution, the contribution of individual basis functions becomes less significant as their values tend to decrease rapidly with distance.

Indeed, as we move away from the training data, most basis functions become close to zero, and the model relies on a small number of them that still have non-negligible values. Consequently, the predictive variance becomes more influenced by the model's assumptions rather than the specifics of the training data. The localized nature of the basis functions implies that the model assigns low weight to data points that are far away, leading to a convergence of predictive variance.

❖ Week 2 – Bayesian Neural Networks

Answers to questions that do not appear in this report can be found in the .ipynb file.

[Question 1.2] We obtain the following figure for Laplace's approximation (left) and the following for MAP estimate (right):



While we obtain the same accuracy in both cases, the main difference is clear: epistemic uncertainty only increases far from training data when using Laplace's approximation. This is exactly what we expected.

[Part I.3] First, we are going to comment the class LinearVariational, explaining each step.

Initialization:

- input_size and output_size determine the dimensions of the layer.
- accumulated_kl_div is a list to accumulate KL-divergence values during training.
- Variational parameters for weights (w_mu and w_rho) and biases (b_mu and b_rho) are initialized as learnable parameters.
- The mean (mu) is initialized to zeros, and the log variance (rho) is initialized to ones.

Sampling (reparameterization trick):

- The sampling method uses the reparameterization trick to sample weights from a Gaussian distribution. For this, we use torch.randn_like() function.

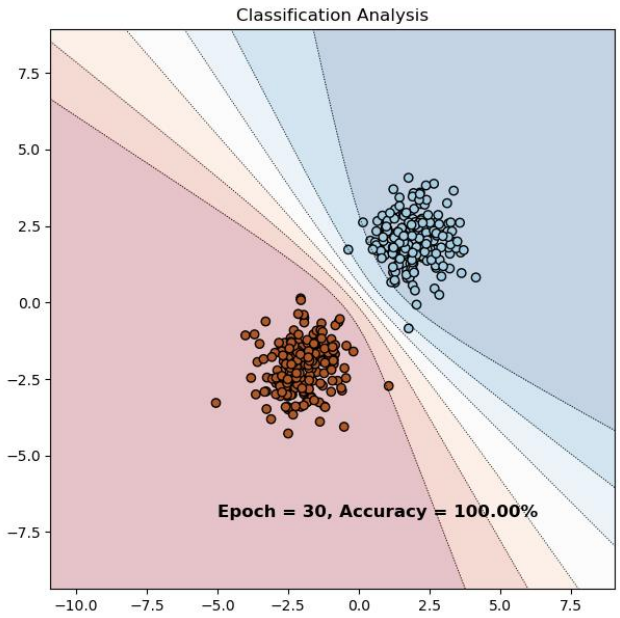
KL-Divergence:

- The kl_divergence method computes the KL-divergence between the variational distribution and the prior distribution.

Forward Pass:

- The forward method performs a forward pass using sampled weights and biases obtained through the reparameterization trick.
- It computes the KL-divergence loss and accumulates it in accumulated_kl_div.

We obtain the following plot for Variational Logistic Regression:



We observe that it stills preserve the accuracy, and it presents more epistemic uncertainty far from training distribution, which is the main difference with MAP estimate.

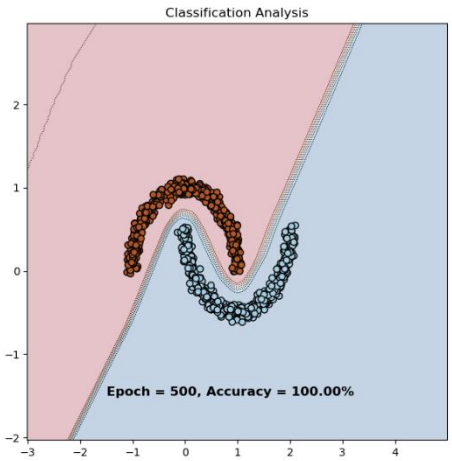
As it is requested, we are going to explain the main differences between Laplace's and VI's approximations (not only in the case of Logistic Regression).

We know that they are both methods used to approximate complex posterior distributions in Bayesian statistics, but they differ in their approaches and underlying principles. We are going to make a table with the main differences:

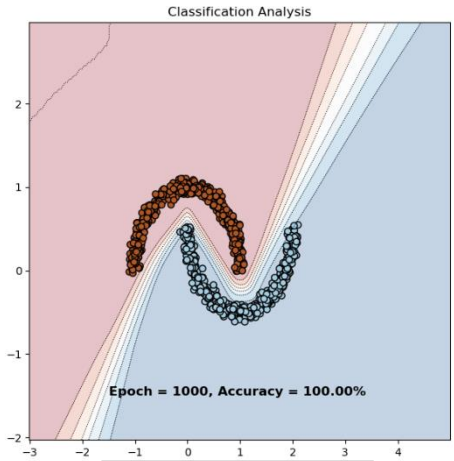
LAPLACE'S APPROXIMATION	VARIATIONAL INFERENCE
Aims to find a Gaussian approximation to the posterior distribution by maximizing the log-posterior at its mode (MAP estimation) and characterizing the curvature of the posterior using the Hessian at that mode.	Formulates the problem as an optimization problem where a simpler, parameterized distribution (variational distribution) is iteratively adjusted to minimize the Kullback-Leibler (KL) divergence from the true posterior.
Provides a point estimate of the posterior distribution, focusing on finding the mode and the curvature at that mode. It does not explicitly model the entire shape of the posterior distribution.	Provides a more flexible framework that allows the variational distribution to capture the full shape of the posterior. It explicitly models the uncertainty in the parameters by using a parameterized distribution.
Involves computing the Hessian of the log-posterior, which can be computationally expensive, especially in high-dimensional spaces.	Generally computationally more scalable, particularly for high-dimensional problems. It involves optimizing the parameters of a variational distribution, which can often be done efficiently using gradient-based optimization methods
Assumes that the posterior distribution is well-approximated by a multivariate Gaussian near its mode.	Allows for more flexibility by choosing a family of distributions to approximate the posterior, making fewer assumptions about the shape of the true posterior.

In summary, Laplace's approximation provides a simple, deterministic point estimate of the posterior, whereas Variational Inference offers a more flexible and probabilistic framework that can capture the entire posterior distribution.

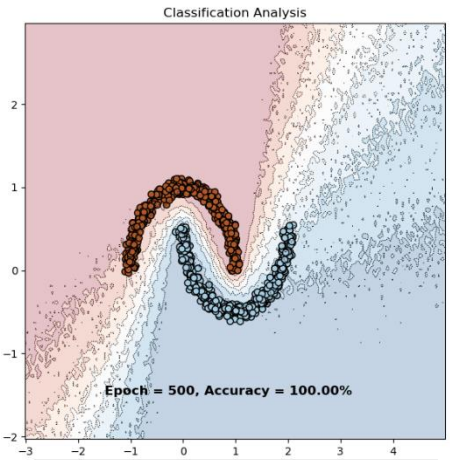
[Question 2.1 + difference] For the two moons dataset, we obtain the following figures:



Deterministic NN



Bayesian NN (VI)



Bayesian NN (MC Dropout)

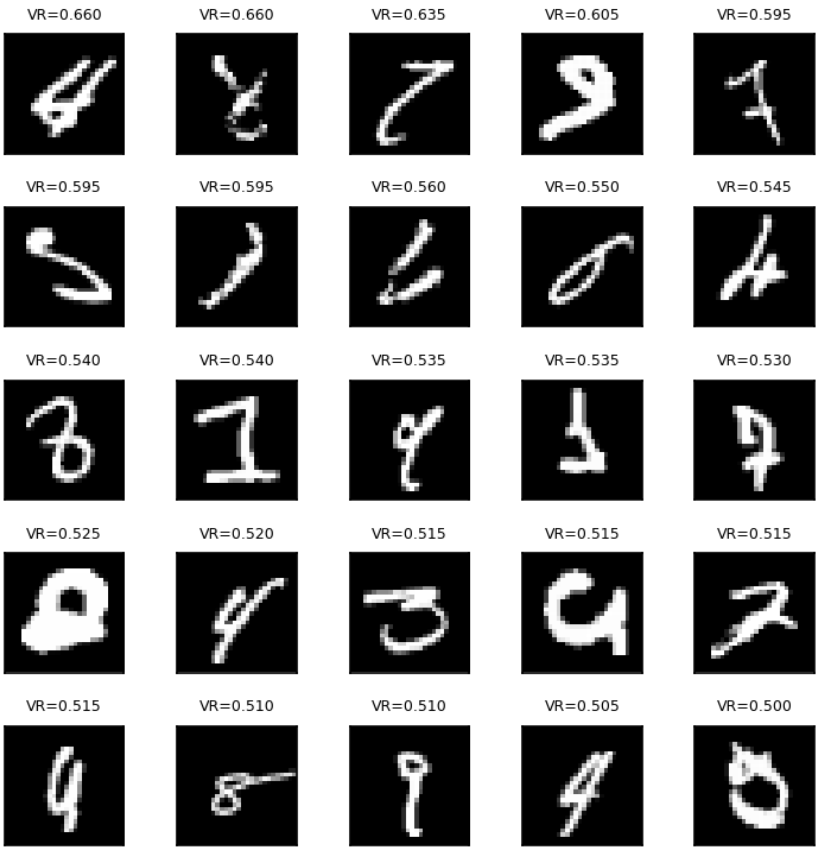
We can see that for Bayesian NN, uncertainty increases far from training area, and this is not the case for deterministic NN. This is exactly what we expected. We also observe a clear difference in the shape of uncertainty between Variational Inference (VI) and Monte Carlo (MC) Dropout.

As requested, we are going to list the main differences between these two techniques:

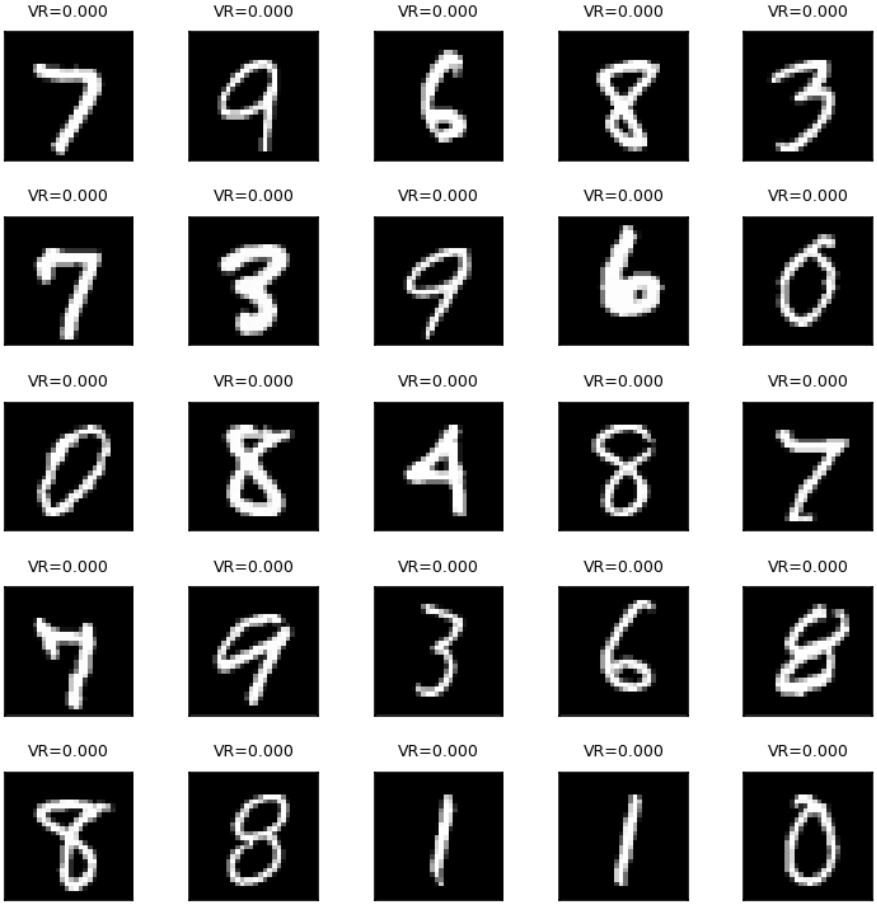
- MC Dropout is a sampling-based approach that utilizes dropout layers, while VI is an optimization-based approach that introduces variational layers (LinearVariational layer).
- MC Dropout is simpler to implement and computationally cheaper, but it may not capture all aspects of uncertainty.
- VI provides a richer characterization of uncertainty. However, it can be computationally more demanding.

❖ Week 3 – Applications of Deep Learning Robustness

[Part I: MC Dropout for classification on MNIST] We first observe that the most confident samples correspond to those with the lowest variation-ratio (VR) , as we expected. In the case of low confidence, we obtain the following images:

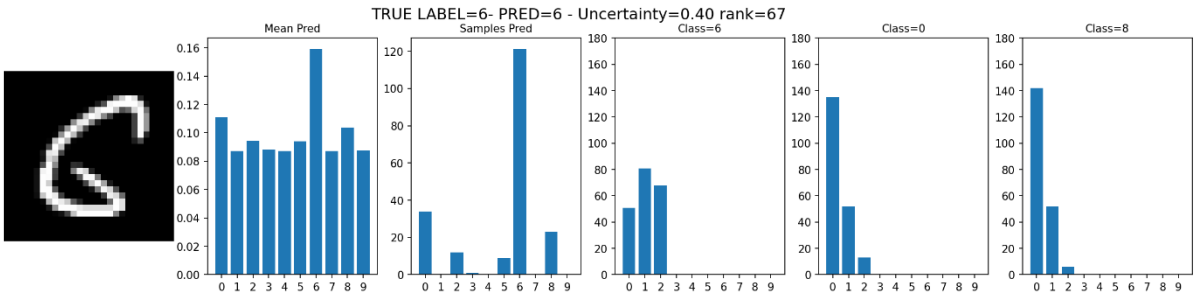


We immediately notice that some of the numbers are not identifiable, which explains why we have low confidence (and therefore a high variation ratio). Let's observe what happens when we increase the confidence.



In this case, numbers are completely identifiable, which is coherent with the fact that uncertainty and VR are low. It is important to note that each training of the model will give slight variations in the images obtained and their associated VR value.

We can use the showUncertainty function to visualize the predictions for a given image. In the following example, we see an image which with our human eye we can say that it corresponds to number 6. We observe that the model does indeed attribute class 6 to the image, although it also tells us that after 6, the most likely classes are 0 and 8.



[Part II: Failure Prediction]

GOAL OF FAILURE PREDICTION:

As seen in class, the goal of failure prediction is to anticipate or identify instances where a system or model is likely to make incorrect predictions or decisions. In various fields such as engineering, finance, healthcare, and more, predicting failures is crucial for improving the reliability, safety, and performance of systems.

COMMENT THE CODE OF LeNetConfidNet CLASS [II.1]:

This class defines a neural network model called LeNetConfidNet, which is based on the LeNet architecture (seen at the beginning of the TP) and is equipped with an auxiliary branch called ConfidNet for uncertainty prediction.

Initialization:

- It sets up the architecture of the neural network. As for LeNet, it includes convolutional layers (conv1 and conv2) and fully connected layers (fc1 and fc2), but this model also includes the layers for the ConfidNet auxiliary branch (uncertainty1 to uncertainty5).
- The model is designed for image classification with a specified number of output classes (n_classes). In our case, n_classes=10 because MNIST dataset has 10 numbers (0 to 9).

Forward Pass:

- MAIN BRANCH: It takes an input tensor x and passes it through the convolutional and fully connected layers. Max pooling is applied after the convolutional layers, and dropout is used for regularization during training. ReLU activation functions are applied after convolutional and fully connected layers, except for the final layer of ConfidNet, which uses a linear activation. This is the same than for LeNet model.
- AUXILIARY BRANCH: It consists of several layers followed by ReLU activation functions. The final layer has a linear activation, producing a single value representing the uncertainty (indeed, we defined uncertainty5 to output a one-dimensional tensor).
- OUTPUTS: The output from the main branch (pred) is the prediction for the input image, and the output from the ConfidNet branch (uncertainty) represents the uncertainty associated with the prediction.

num_flat_features:

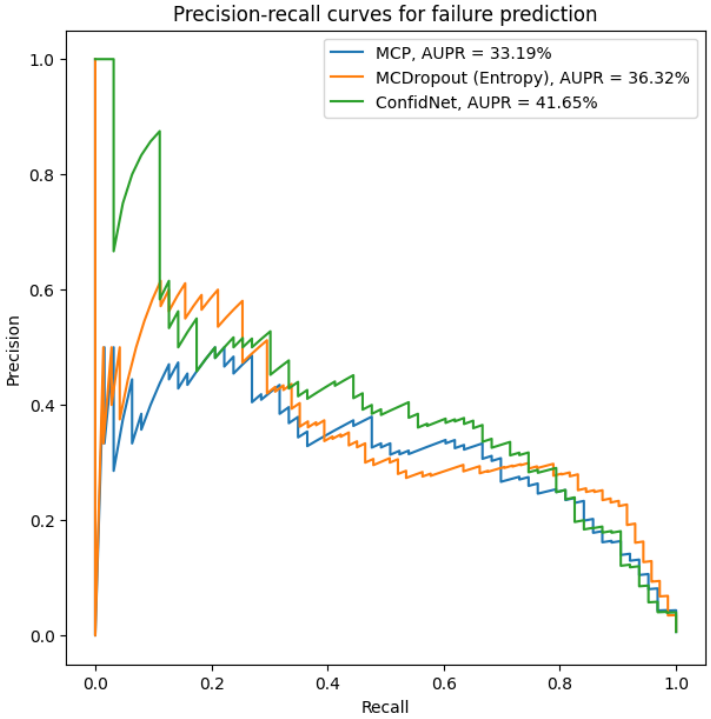
- This method calculates the number of features in a flattened tensor, excluding the batch dimension.

count_parameters:

- This method counts the total number of trainable parameters in the model.

ANALYSE RESULTS BETWEEN MCP, MCDropout AND CondifNet [II.2]:

The goal of this part is to evaluate failure prediction performances by observing precision-recall curves and AUPR. Here, I observed that the figure obtained varies significantly with each training of the model. We are going to present one output:



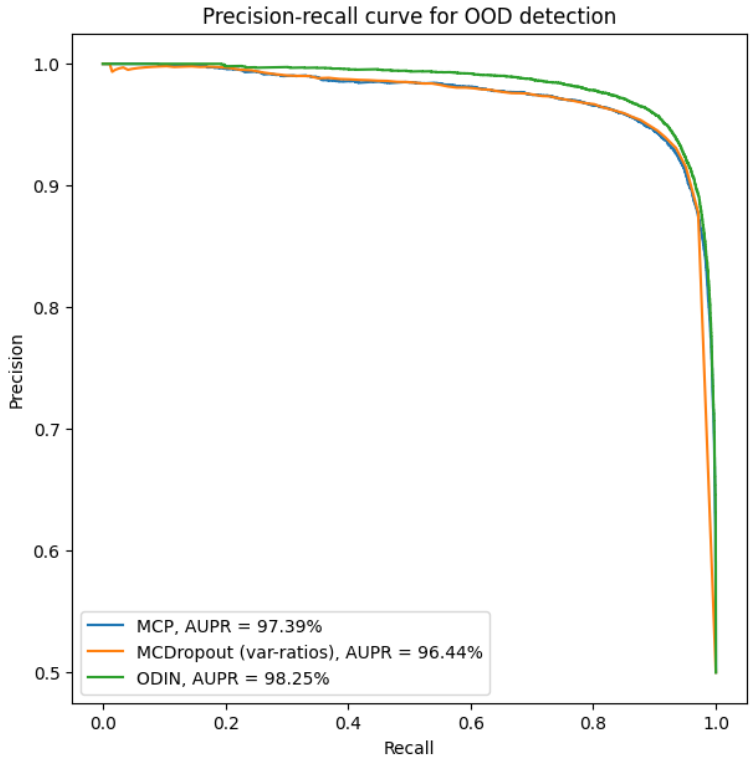
In this case, we observe that ConfidNet performs the best. Indeed, it provides higher precision across different levels of recall, and it presents the higher AUPR.

[Part III: Out-of-distribution detection] As requested, we are going to analyse results and explain the difference between the 3 methods (MCP, MC Dropout, ODIN) in terms of out-of-distribution detection (OOD). For this, we are going to make a table (as we did to compare Laplace's approximation and Variational Inference):

MCP	MC Dropout	ODIN
MCP is a measure of confidence in a model's prediction. It is the highest probability assigned to any class by the softmax function.	MC Dropout is a Bayesian approximation method that leverages dropout to estimate model uncertainty.	ODIN is a threshold-based detector enhancing maximum softmax probabilities with two extensions.
Given an input, the model's output probabilities are obtained using softmax, and MCP is the maximum probability among these.	During inference, the model is run multiple times with dropout enabled, resulting in an ensemble of predictions.	ODIN introduces temperature scaling during inference, effectively sharpening the softmax probabilities. Higher temperature values increase the gap between the maximum and other probabilities. It also applies an adversarial perturbation to the input data to make the model more

		sensitive to in-distribution samples.
MCP does not consider uncertainty in the model's parameters or the input data. It is solely based on the probability distribution of the predicted classes.	MC Dropout provides a measure of uncertainty by considering the variability in predictions across different dropout samples.	ODIN aims to increase the margin between in-distribution and out-of-distribution samples by adjusting softmax probabilities and introducing an adversarial perturbation. The model's confidence is enhanced for in-distribution samples.

We can also compare the obtained precision-recall curves of each OOD method along with their AUPR values:



We observe that ODIN method performs the best. Indeed, it provides higher precision across different levels of recall, and it presents the higher AUPR.