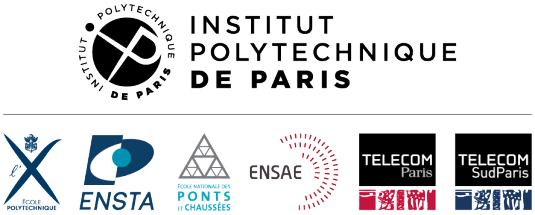
Bayesian Statistics Project
Paper : "*Deep Learning : A Bayesian Perspective*"

ELMIMOUNI Zakarya
ENSAE Paris
zakarya.elmimouni@ensae.fr

KHAIRALDIN Ahmed
ENSAE Paris
ahmed.khairaldin@ensae.fr

RAZIG Amine
ENSAE Paris
amine.razig@ensae.fr



Project summary

This applied project is based on the foundational academic article *"Deep Learning: A Bayesian Perspective"*Nicholas Polson [2017]. The objective is to explore this statistical application, by connecting theoretical insights from the paper with practical data analysis, the project aims to evaluate the applicability and effectiveness of Bayesian approaches in real-world scenarios.

The project is structured around three core elements. First, we summarize the key objectives and contributions of Nicholas Polson [2017], focusing on the way of integrating Bayesian methods with deep learning models. Then selected datasets will serve as a basis for applying Bayesian methods. We will conduct a detailed analysis by implementing techniques and models aligned with the concepts explored in the paper and class materials. All our experiments are available in this Github repository[1], where you can find the code, some theoretical explanations and data visualization.

1 Introduction

On one hand, we want to highlight that this project is a Bayesian Statistics project. Bayesian statistics provide a framework for learning from data through the principle of probability. One of the foundations of this statistical field is Bayes' rule, which we will describe later. Essentially, it is a structured way of updating beliefs about unknown parameters using observed data, integrating both prior knowledge and new information. This approach is interesting because it aligns with the principles of rational decision-making.

On the other hand, Deep learning, has revolutionized machine learning by enabling models to learn complex, non-linear relationships from data. Multi-layers Perceptrons are composed of layers of

[1] https://github.com/ahmedkakiAK/Bayesian_stats_project.git

interconnected neurons, each applying weights and learned activation functions to extract features and make predictions. While traditional deep learning often focuses on deterministic parameter estimates, a Bayesian perspective introduces probabilistic modeling, where uncertainty in parameters and predictions is explicitly quantified. By incorporating a priori weights and using Bayesian inference. The bayesian deep learning framework can improve robustness, provide uncertainty estimates for predictions and help prevent over-fitting, which is a fundamental concept in an applied ML project.

In this project, we decided to tackle a medical and energy questions with deep learning from a Bayesian perspective approach using two datasets: the ***National Institute of Diabetes and Digestive and Kidney Diseases' diabetes dataset***, and the ***regional transmission organization PJM's hourly energy consumption data***. First, the idea is to apply such a framework we want to show applications in different fields where uncertainty quantification is important. In medical applications for example, Bayesian methods enable to optimize a treatment strategies by explicitly accounting for uncertainty. This ensures that decisions are both driven by the data and tailored to individual beliefs, even when information is incomplete.

2 Presentation of the dataset

2.1 Description

The first dataset, from the **National Institute of Diabetes and Digestive and Kidney Diseases**, contains health-related information aimed at predicting the onset of diabetes. It includes 768 entries and 9 characteristics, collected from patients who have undergone medical examinations. The dataset includes glucose level, blood pressure or teh age for instance.

For the second one, **PJM hourly energy consumption data provides detailed information on electricity consumption** in the regions managed by the PJM (Pennsylvania-New Jersey-Maryland) regional transmission organization. This dataset includes hourly energy consumption data, providing an overview of energy demand in different sectors.

2.2 Objectives

In this project, we have chosen to explore two distinct domains. The first objective is to use the diabetes dataset to evaluate uncertainty about developing diabetes, by exploiting the various medical variables available. The second objective is to analyze PJM's hourly energy consumption data to predict fluctuations in energy demand. We will use Bayesian framework to capture uncertainty and improve model robustness. With these two datasets, we aim to demonstrate the effectiveness of Bayesian models in deep learning for prediction and uncertainty.

3 Modelling

Modeling is the focus of our report and requires the explanation of several concepts such as deep learning, Bayesian inference, Monte Carlo estimation methods, and Dropout approximation. These are concepts that we will present to justify our model (Monte Carlo Dropout) used on real data. We call Monte Carlo Dropout an MLP with Dropout layers and several forward passes for prediction to model uncertainty.

3.1 Theoretical basis of MLP

The first concept we will explore is neural networks. There are many different neural network architectures, varying in complexity and suitability for specific tasks. Here, we focus on a simple architecture developed in Nicholas Polson [2017]: the multi-layer perceptron (MLP).

The basic idea behind the MLP is that it consists of several layers of neurons. Each neuron corresponds to a linear combination of the neurons in the previous layer. A non-linear function, known as the activation function, is then applied to this linear combination. This structure allows the model to capture complex, non-linear relationships between inputs and outputs, which is essential for solving

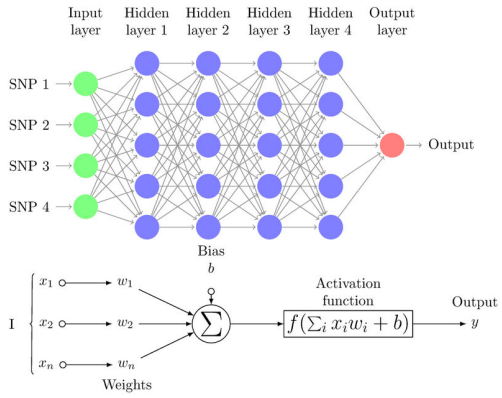


Figure 1: Illustration of the strcuture of a Neural Network. Source: Miguel Perez-Enciso.

more intricate problems that linear models cannot handle. The model produces a prediction, denoted as $\hat{Y}_{W,b}(X_i)$, which is essentially a composition of the different layers in the network.

However, while MLPs can learn powerful models from data, they often suffer from a critical problem: the uncertainty of model predictions. In traditional MLPs, the network learns point estimates for weights $W^{(l)}$ for all layers l, which may not reflect the underlying uncertainty of the model or data. This is particularly problematic in applications where understanding the uncertainty of the prediction is important, such as medical diagnostics. Since the main subject of this academic project is not deep learning we do not develop much about the mathematical formalization of a neural network structure, forward and backward, but some details are given in the appendix.

3.2 Bayesian Inferance

So, this is where Bayesian inference comes into play. Bayesian methods allow us to treat the weights of a neural network as random variables rather than fixed parameters. Instead of learning a single set of weights, the network learns a distribution over weights, which reflects the uncertainty about their true values. By incorporating Bayesian inference, we can not only make predictions but also quantify the uncertainty associated with those predictions.

But first, let's introduce some basic concepts.

3.2.1 Bayes for neural networks

For us the parameters to estimate are the weights of the network. Bayes rule incorporate prior knowledge about an unknown θ before the observation of the data X. It is given by:

$$p(\theta \mid X) = \frac{p(X \mid \theta)p(\theta)}{\int p(X \mid \theta)p(\theta)d\theta}$$

This is interpretable in the way that with a given prior, the posterior has all information about the parameter θ based on the data X. So, for the task of predicting y' with x', the parameter is still θ and the data is $D = (X, Y)$ and so we have :

$$p(\theta \mid X, Y) = \frac{p(Y \mid \theta, X)p(\theta)}{\underbrace{\int p(x \mid \theta, Y)p(\theta)\, d\theta}_{\text{not tractable}}}$$

$$p(y' \mid x', X, Y) = \int p(y' \mid x', \theta)p(\theta \mid X, Y)\, d\theta$$

With the total probability formula we see that Bayes predictor averages over all of the models parameterized by θ. But a it is notified, some parts are not tractable. We can use MCMC usualy but because of the numbers of parameters this approach is difficult. Instead, Nicholas Polson [2017] explain the following approach. insated of finding the posterior distribution of the parameters $p(\theta \mid X, Y)$ we find an approximation of this distribution denoted q. For that we minimize the Kullback-Leibler divergence between $q(\theta|\phi)$ and $p(\theta|D)$

3.2.2 Variational Inference

The goal in variational inference is to minimize the KL divergence. However, directly minimizing the KL divergence involves the intractable term $\log p(\theta|D)$. To address this, the paper used variational approaches instead of MCMC methods, which approximate $p(\theta|D)$ with a simpler distribution $q(\theta|\phi)$, where ϕ are the parameters of the approximation. To address this, we use the following useful identity (details of the proof in appendix) :

$$\log p(D) = \text{ELBO}(\phi) + \text{KL}(q \parallel p),$$

where:

- $\text{ELBO}(\phi)$ is the Evidence Lower Bound.
- $\text{KL}(q \parallel p)$ represents the KL divergence.

Since the sum does not depend on ϕ, minimizing $\text{KL}(q \parallel p)$ is equivalent to maximizing the ELBO:

$$\text{ELBO}(\phi) = \int q(\theta \mid D, \phi) \log \frac{p(Y \mid X, \theta) p(\theta)}{q(\theta \mid D, \phi)} \, d\theta.$$

Thus, maximizing the ELBO can be solved using stochastic gradient descent with respect to ϕ. This approach allows us to approximate the posterior while avoiding the intractable $\log p(\theta \mid D)$.

3.2.3 Reparametrization for Monte Carlo method

Maximizing the ELBO is usually done with Gradient Descent. However computing these gradients remains a challenge, beaucause the gradient of the first term $\nabla_\phi \int q(\theta|D, \phi) \log p(Y|X, \theta) d\theta = \nabla_\phi E_q \log p(Y|X, \theta)$ is not an expectation anymore and thus cannot be calculated using Monte Carlo methods. To overcome this, two primary techniques are used.

First, the log-derivative trick, uses the following identity :

$$\nabla_\phi Eq[\log p(Y|X, \theta)] = E_q[\nabla\phi \log q(\theta|\phi) \cdot \log p(Y|X, \theta)].$$

This method is particularly useful for simple distributions $q(\theta|\phi)$, where derivatives and samples are easy to obtain.

Second, the **reparametrization** trick redefines the parameters θ as a deterministic transformation of an independent random variable ϵ, enabling Monte Carlo-based gradient estimation. If we represent:

$$\theta = g(\epsilon, \phi) \quad \text{where} \quad \epsilon \sim r(\epsilon),$$

then the gradient can be re-expressed as:

$$\nabla_\phi E_q[\log p(Y \mid X, \theta)] = E_\epsilon[\nabla_\phi g(\epsilon, \phi) \cdot \nabla_\theta \log p(Y \mid X, \theta)].$$

This formulation ensures that gradients can be estimated using Monte Carlo methods by sampling ϵ from its known distribution $r(\epsilon)$.

MCMC Framework:

In Bayesian inference, Markov Chain Monte Carlo (MCMC) is a common method for simulating samples from a target posterior distribution. The principle involves generating a sequence of random variables $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(R)}\}$ such that the distribution of $\theta^{(r)}$ approximates the posterior $p(\theta \mid D)$ as $r \to \infty$.

To incorporate the reparametrization trick into the MCMC framework, the transformation $\theta = g(\epsilon, \phi)$ is used in each step of the Markov chain. The sampling procedure becomes:

$$\epsilon^{(r)} \sim r(\epsilon), \quad \theta^{(r)} = g(\epsilon^{(r)}, \phi).$$

Algorithm 1 MCMC Method with Reparametrization

Input: Initial state $\theta^{(0)}$, reparametrization function $g(\epsilon, \phi)$, independent distribution $r(\epsilon)$, number of iterations R.
Step 1: Initialization
Set the starting point $\theta^{(0)}$.
Step 2: Iterative Transition
for $r = 1$ to R **do**
    Sample $\epsilon^{(r)} \sim r(\epsilon)$.
    Compute $\theta^{(r)} = g(\epsilon^{(r)}, \phi)$.
end for
Output: Sequence $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(R)}\}$.
Convergence:
Under appropriate conditions on $g(\cdot)$ and $r(\cdot)$, the distribution of $\theta^{(r)} \mid \theta^{(0)}$ converges to a stationary distribution as $r \to \infty$.

The resulting sequence $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(R)}\}$ converges to a stationary distribution that incorporates the effects of the reparametrization.

Thus, the gradient estimation becomes feasible, as the dependence on ϕ is explicitly captured through the deterministic transformation $g(\epsilon, \phi)$.

3.3 Dropout approximation

Dropout is introduced as a regularization technique, it can be interpreted as a Bayesian approximation. Specifically, during training, dropout randomly masks (sets to zero) some neurons, simulating a stochastic model ensemble. During inference, by keeping dropout active and performing multiple forward passes with different masks, we passe the same input many times (N samples) in the network and we collect N different predictions, these predcitions are used to approximate the mean $E[y^*|x^*] \approx \frac{1}{N}\sum_{i=1}^{N} \hat{y}_i$ and the variance $\text{Var}[y^*|x^*] \approx \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - E[y^*|x^*])^2$. The predictions generated by the deep learning models with dropout are nothing but samples from predictive posterior distribution. This interpretation was formalized in the work "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning", Yarin Gal [2016]. They showed that dropout can be seen as a bayesian approximation. more specifically, applicating Dropout during the training and test is equivalent to appoximate a baeysian model with a distibution of weights.

Therefore, in what follows, we use the Monte Carlo Dropout model as an approximation of a Bayesian Neural Network with variational inference.

4 Results

In Nicholas Polson [2017], the experiment involved a simple MLP applied to a ranking task without any Bayesian deep learning application.

In this section, we apply our model: an MLP with dropout layers to interpret uncertainty using a Monte Carlo technique through multiple forward passes of the model. To achieve this, we activate the dropout layers during both training and evaluation.

First, we use this model for a binary classification task on the diabetes dataset, as well as on a simple toy dataset we created to facilitate better interpretability.

Next, we apply our model to a time-series dataset: the hourly estimated power consumption provided by the PJM organization in the eastern United States, spanning from 2002 to 2018. We plot the confidence intervals and the Monte Carlo mean of the forward pass predictions and compare them with ground-truth values.

4.1 Binary Classification

The binary classification task involved predicting an outcome variable: whether a person has diabetes or not, using the two most important features of the dataset (selected with XGBoost and

correlations): Glucose level and Body Mass Index (BMI). These features were chosen to enable the plotting of decision boundaries in a 2D space.

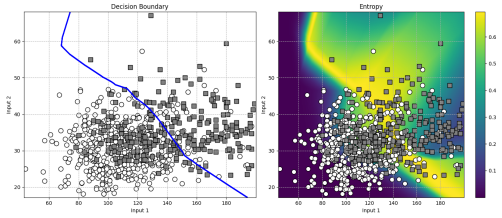
Model Parameters. The model we trained consisted of two hidden layers (with 64 and 32 hidden units each) with ReLU activation functions and dropout applied after each layer. After fine-tuning, we selected a dropout rate of 0.2. The optimizer used for this task was Adam with a learning rate of 0.01. We trained the model for 2000 epochs. Overfitting was not a concern in this case, as the inference was conducted on the training data for experimental purposes.

We first plotted the decision boundary and entropy from a single forward pass on the training set without dropout. The results are shown in the figure2a. For our binary classification task, entropy was calculated based on the predicted probabilities for each class. When the model is uncertain (i.e., when the probabilities are close to 0.5), entropy is high. Conversely, when the model is confident (i.e., when the probabilities are close to 0 or 1), entropy is low. High entropy indicates greater uncertainty, while low entropy reflects higher confidence in the predictions. However, this single forward pass is insufficient to fully interpret uncertainty, as the model in this case is deterministic.

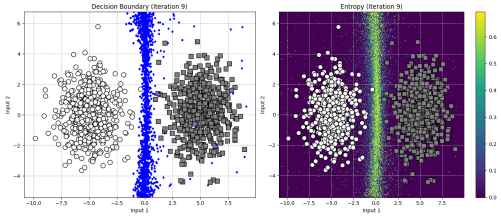
To address this, we performed multiple forward passes using Monte Carlo (MC) Dropout to better interpret uncertainty at each pass. However, due to the noise introduced by dropout and the complexity of our dataset (as detailed in the notebook), we were unable to achieve satisfactory results in terms of decision boundaries. Therefore, we conducted the same experiment on a toy dataset consisting of two linearly separable classes in a 2D space, which allowed us to better interpret uncertainty through decision boundaries and entropy.

In the figure2b, we observe the results of one of the forward passes. Compared to the other plot, there is more noise due to the dropout. However, this allows us to interpret uncertainty. Zones where the decision boundary is narrow (with respect to the 0-ordinate axis) in most iterations correspond to areas with the least uncertainty, whereas zones where the boundary is wider indicate regions where the model is most uncertain.

As shown in the entropy plot, dropout introduces additional noise, which can cause yellow points (representing areas of highest uncertainty) to appear in different parts of the grid as anomalies in the decision boundary, scattered across the grid. In these zones, these anomalies cannot be fully attributed to uncertainty, but rather to the noise introduced by dropout.



(a) Decision boundary and entropy a simple prediction with the MLP (no Dropout for evaluation) for diabetes dataset



(b) Decision Boundary and entropy for one of the forward passes of MC Dropout (dropout activated for evaluation) for toy dataset

Figure 2: Decision boundaries and entropies plot for both datasets: diabetes and toy dataset

4.2 Time-series Regression

We also applied our model to a regression task on the hourly power consumption dataset, PJME.

Model Parameters. We used an MC Dropout MLP with 4 hidden layers (containing 1000, 2000, 2000, and 1000 units, respectively) and SELU activation functions. Dropout was applied after each layer with a dropout rate of 0.4. The model was trained for 200 epochs with Early Stopping, which halted training after 16 epochs. The optimizer used was the NAdam optimizer with a default learning rate of 0.001.

After training, we performed several forward passes (with dropout enabled) on the test set, which included observations from 2015 to 2018. We plotted in figure3 the MC mean predictions along with the confidence intervals, which provide an interpretation of the model's uncertainty. It is important

to note that an MLP is not ideally suited for time series data due to the inherent time dependencies and seasonal patterns in such datasets.

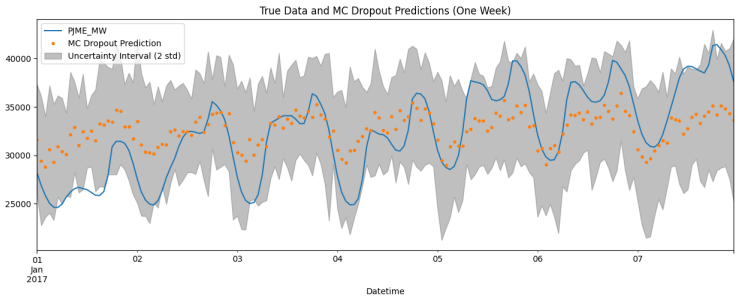


Figure 3: Mean predictions and confidence intervals compared with ground-truth values in one week of the test set.

5 Conclusion

In Nicholas Polson [2017], authors explores a method using weight distributions, showing that applying Dropout during both training and testing is equivalent to approximating a Bayesian NN with variational inference Yarin Gal [2016]. Experiments were conducted on the PJME hourly energy consumption, a time series, where we were able to interpret uncertainty through confidence intervals applied to our predictions. The results obtained, however, were not particularly strong. This can be attributed to the fact that (MLPs), as an architecture, are not designed to handle time series data effectively. Thus, a suggested improvement would be to experiment bayesian techniques (dropout or variational inference) with Recurrent Neural Networks (RNNs) instead of MLPs, since they are more suited to capturing temporal dependencies, which could lead to better modeling of uncertainty and more accurate predictions (smaller confidence intervals).

GitHub

Instructions to run the code are available in the README file of the following github:

GitHub Repository link : $https://github.com/ahmedkakiAK/Bayesian\_stats\_project.git$

References

Vadim Sokolov Nicholas Polson. Deep learning: A bayesian perspective. 12:1275–1304, 2017.

Zoubin Ghahramani Yarin Gal. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. 2016.

Appendix

MLP

Forward Propagation

The goal of forward propagation is to compute the prediction $\hat{y}$ and the loss $\mathcal{L}$ using the current model parameters $(w^{[l]}, b^{[l]})$. The neural network receives an input x. For layer l, the linear output is:

$$z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}$$

where $a^{[0]} = x$ is the input to the first layer. An activation function $f(\cdot)$ is applied to produce:

$$a^{[l]} = f(z^{[l]})$$

And then, the last layer produces the prediction $\hat{y}$.

Backward Propagation

The purpose of backward propagation is to compute the **gradients** of the loss with respect to the model parameters $(w^{[l]}, b^{[l]})$ to optimize them. We start with the derivative of the loss with respect to the predicted output:

$$\frac{\partial \mathcal{L}}{\partial \hat{y}}$$

and then, propagate this derivative backward through the network, using the cahin rule, to compute:

$$\frac{\partial \mathcal{L}}{\partial z^{[l]}}, \quad \frac{\partial \mathcal{L}}{\partial w^{[l]}}, \quad \frac{\partial \mathcal{L}}{\partial b^{[l]}}$$

Update of the parameters simply use the gradient descent rule.

MSE loss

If we consider a given training dataset

$$D = \{(Y^{(i)}, X^{(i)})\}_{i=1}^{T}$$

of input-output pairs and a loss function $\mathcal{L}(Y, \hat{Y})$, we compute:

$$\hat{W} = (\hat{W}_0, \ldots, \hat{W}_L) \quad \text{and} \quad \hat{b} = (\hat{b}_0, \ldots, \hat{b}_L)$$

by solving:

$$\arg\min_{W,b} \frac{1}{T} \sum_{i=1}^{T} \mathcal{L}\left(Y_i, \hat{Y}_{W,b}(X_i)\right).$$

For the L_2-norm in a traditional least squares setting:

$$\mathcal{L}(Y_i, \hat{Y}(X_i)) = \|Y_i - \hat{Y}(X_i)\|_2^2,$$

the target function becomes the mean-squared error (MSE).

Derivation of $\log p(D)$

The goal is to express $\log p(D)$ in terms of the Evidence Lower Bound and the KL divergence. We start by introducing a variational distribution $q(\theta \mid D, \phi)$ to approximate the true posterior $p(\theta \mid D)$.

First, the marginal likelihood $p(D)$ can be written as:

$$\log p(D) = \log \int p(\theta, D)\, d\theta,$$

where $p(\theta, D) = p(\theta)p(D \mid \theta)$. By introducing the variational distribution $q(\theta \mid D, \phi)$, we rewrite:

$$\log p(D) = \log \int q(\theta \mid D, \phi) \frac{p(\theta, D)}{q(\theta \mid D, \phi)}\, d\theta.$$

Using Jensen's inequality ($\log E[X] \geq E[\log X]$), we lower-bound $\log p(D)$:

$$\log p(D) \geq \int q(\theta \mid D, \phi) \log \frac{p(\theta, D)}{q(\theta \mid D, \phi)} \, d\theta.$$

The term on the right-hand side is the Evidence Lower Bound :

$$\text{ELBO}(\phi) = \int q(\theta \mid D, \phi) \log \frac{p(\theta, D)}{q(\theta \mid D, \phi)} \, d\theta.$$

Rewriting $\log p(D)$ exactly, without Jensen's inequality:

$$\log p(D) = \int q(\theta \mid D, \phi) \log \frac{p(\theta, D)}{q(\theta \mid D, \phi)} \, d\theta + \int q(\theta \mid D, \phi) \log \frac{q(\theta \mid D, \phi)}{p(\theta \mid D)} \, d\theta.$$

The first term is the ELBO and the second term is the Kullback-Leibler (KL) divergence between $q(\theta \mid D, \phi)$ and the true posterior $p(\theta \mid D)$:

$$\text{KL}(q \parallel p) = \int q(\theta \mid D, \phi) \log \frac{q(\theta \mid D, \phi)}{p(\theta \mid D)} \, d\theta.$$

Thus, we can write:

$$\log p(D) = \text{ELBO}(\phi) + \text{KL}(q \parallel p).$$

The KL divergence $\text{KL}(q \parallel p)$ is always non-negative. Therefore:

$$\log p(D) \geq \text{ELBO}(\phi).$$

So to conclude, maximizing the ELBO $\text{ELBO}(\phi)$ with respect to the variational parameters ϕ reduces the KL divergence $\text{KL}(q \parallel p)$ and approximates the true posterior $p(\theta \mid D)$. The decomposition is given by:

$$\log p(D) = \text{ELBO}(\phi) + \text{KL}(q \parallel p).$$