© Revista de Matemática: Teoría y Aplicaciones 2022 **29**(2) : 289–312
cimpa – ucr issn: 1409-2433 (Print), 2215-3373 (Online)
doi: https://doi.org/10.15517/rmta.v29i2.48885

DEEP GAUSSIAN PROCESSES AND INFINITE NEURAL NETWORKS FOR THE ANALYSIS OF EEG SIGNALS IN ALZHEIMER'S DISEASES

PROCESOS GAUSIANOS PROFUNDOS Y REDES NEURONALES INFINITAS PARA EL ANÁLISIS DE SEÑALES EEG EN LA ENFERMEDAD DE ALZHEIMER

Krishna Román* Andy Cumbicus† Saba Infante‡
Rigoberto Fonseca-Delgado§

Received: 1/Nov/2021; Revised: 23/Jun/2022;
Accepted: 28/Jun/2022

Revista de Matemática: Teoría y Aplicaciones is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.
http://creativecommons.org/licenses/by-nc-sa/4.0/

*Yachay Tech University, School of Mathematical and Computational Sciences, Urcuquí, Ecuador. E-mail: krishna.roman@yachaytech.edu.ec
†Misma dirección que/*Same address as*: K.Román. E-mail: andy.cumbicus@yachaytech.edu.ec
‡Misma dirección que/*Same address as*: K. Román. E-mail: sinfante@yachaytech.edu.ec
§Misma dirección que/*Same address as*: K. Román. E-mail: rfonseca@yachaytech.edu.ec

Abstract

Deep neural network models (DGPs) can be represented hierarchically by a sequential composition of layers. When the prior distribution over the weights and biases are independently identically distributed, there is an equivalence with Gaussian processes (GP) in the limit of an infinite network width. DGPs are non-parametric statistical models used to characterize patterns of complex non-linear systems due to their flexibility, greater generalization capacity, and a natural way of making inferences about the parameters and states of the system. This article proposes a hierarchical Bayesian structure to model the weights and biases of a deep neural network. We deduce a general formula to calculate the integrals of Gaussian processes with non-linear transfer densities and obtain a kernel to estimate the covariance functions. In the methodology, we conduct an empirical study analyzing an electroencephalogram (EEG) database for diagnosing Alzheimer's disease. Additionally, the DGPs models are estimated and compared with the NN models for 5, 10, 50, 100, 500, and 1000 neurons in the hidden layer, considering two transfer functions: Rectified Linear Unit (ReLU) and hyperbolic Tangent (Tanh). The results show good performance in the classification of the signals. Finally, we use the mean square error as a goodness of fit measure to validate the proposed models, obtaining low estimation errors.

Keywords: deep Gaussian process; Alzheimer disease; electroencephalogram.

Resumen

Los modelos de redes neuronales profundos (DGPs) se pueden representar jerárquicamente mediante una composición secuencial de capas. Cuando la distribución prior sobre los pesos y sesgos son independientes idénticamente distribuidos, existe una equivalencia con los procesos Gaussiano (GP), en el límite de una anchura de red infinita. Los DGPs son modelos estadísticos no paramétricos y se utilizan para caracterizar los patrones de sistema no lineales complejos, por su flexibilidad, mayor capacidad de generalización, y porque proporcionan una forma natural para hacer inferencia sobre los parámetros y estados del sistema. En este artículo se propone una estructura Bayesiana jerárquica para modelar los pesos y sesgos de la red neuronal profunda, se deduce una formula general para calcular las integrales de procesos Gaussianos con funciones de transferencias no lineles, y se obtiene un núcleo para estimar las funciones de covarianzas. Para ilustrar la metodología se realiza un estudio empírico analizando una base de datos de electroencefalogramas (EEG) para el diagnóstico de la enfermedad de Alzheimer. Adicionalmente, se estiman los modelos DGPs, y se comparan con los modelos de NN para 5, 10, 50, 100, 500 y 1000 neuronas en la capa oculta, considerando dos funciones de transferencia: Unidad Lineal Rectificada (ReLU) y tangenge hiperbólica

(Tanh). Los resultados demuestran buen desempeño en la clasificación de las señales. Finalmente, utilizó como medida de bondad de ajuste el error cuadrático medio para validar los modelos propuestos, obteniéndose errores de estimación bajos.

Palabras clave: procesos gausianos profundos; enfermedad de Alzheimer; electroencefalogramas.

Mathematics Subject Classification: 60G15, 60H35.

1 Introduction

Deep machine learning systems are computational algorithms that provide powerful modern tools that allow the use of mathematical models with complex structures by including multiple intermediate layers at different levels combined with transfer functions with non-linear systems.

Neural networks are machine learning models that have received a lot of attention in recent years due to their success in many real-world applications: they have been used very frequently in filtering content on social networks, in natural language processing, pattern recognition in Big data, tracking objects in a sequence of images or videos, face and voice recognition, human mobility, mass information dissemination in networks, electronic commerce and classification of relevant information, among many other applications, LeCun et al. (2015) [13], Goodfellow et al. (2016) [8], Mosavi et al. (2020) [20], and Luca et al. (2020) [15].

In this work, a combined technique of Gaussian processes (GP) with neural network models is used, following the approach outlined by: MacKay (1992) [16], MacKay (1995) [**?**], Neal (1996) [21], and Williams (1996) [27]. GPs are used to model functional data because they are flexible, robust to outliers, and estimate calibrated uncertainty. Deep Gaussian Processes (DGP) are a generalization of a multilayer neural network viewed as a GP in the limit width. Salimbeni and Deisenroth (2017) [**?**] used an inference algorithm variational doubly stochastic algorithm that does not force independence between layers. Also, they demonstrated that a DGP model could be used effectively for many data points. They provide strong empirical evidence of the inference scheme for DGPs and show that they work well in practice in both classification and regression. Zhao et al. (2021) [30] used a state-space model for the deep Gaussian process (DGP) regression. They constructed a DGP by placing a GP prior transformed into the length and magnitude scales at each hierarchy level. They used a posterior maximum estimation procedure based on filtering algorithms and demonstrated the performance using non-stationary synthetic and gravitational wave signals.

Wilson and Izmailov (2020) [28] showed that deep ensembles effectively estimate the marginal distributions. They also investigated priors on the considered functions by defining vague priors on the neural network weights. In addition, they demonstrate properties, generalize the models from a probabilistic perspective, and obtain results equivalent to those reproduced using a GP. Lee et al. (2018) [14] demonstrated an equivalence between infinitely wide deep networks and GPs, developed computationally efficient methods to compute the covariance function of GPs and connected GPs with the theory of signal propagation in random neural networks. Investigations in the same direction highlight the work of Schoenholz et al. (2017) [26], Matthews et al. (2018) [18], Novak et al. (2019) [22], Garriga-Alonso, et al. (2019) [**?**], Agrawal et al. (2020) [2], Damianou and Lawrence (2013) [6], Khan et al. (2019) [11], Hazan and Jaakkola, (2015) [9]. We can mention related works to these topics: Infante et al. (2008) [10], Cedeño et al. (2021.a) [3] and Cedeño et al. (2021.b) [4].

The contribution in this work is based on the proposal of a hierarchical Bayesian structure to model the weights and biases of a neural network with infinite width. We established a general formula to calculate the integrals of Gaussian processes for sigmoid functions with non-linear structures, and we obtained a kernel to update the covariance functions. Finally, we made an application on a set of electroencephalogram signals from patients with Alzheimer's disease.

The rest of the article is as follows: the Deep Neural Networks are defined in Section 2, the Gaussian processes are described in Section 3, the connection between the Deep Neural Networks and Gaussian Processes is made in Section 4, and the results are shown in the Section 5, and finally some discussions and conclusions are established in the Section 6.

2 Deep neural networks

Mathematically a neural network can be defined as a directed graph with vertices representing neurons and edges expressing connections. Each neuron's input is a weighted sum of the output of all previous layer's neurons connected to the input. There are many variants of neural networks that differ by their architecture. The simplest of these forms is the forward neural network, also known as the feedforward neural network.

Deep neural networks compose computations performed by many layers. Denoting the output of hidden layers by $h_i^{(l)}$, where $l \in \{1, \ldots, L\}$, $i \in \{1, \ldots, N_l\}$ denote the indices of the neuron within the layer that receives the information from the neurons of the previous layer $h_j^{(l-1)}$, $j \in \{1, \ldots, N_{l-1}\}$.

The output j represented by the $j-$th neuron in the output layer, is connected to the input vector x via a biased weighted sum and an non-linear activation function ϕ. The $j-$th component of the network output, $h_j^{(l)}$, is computed as:

$$h_j^{(l)}(x) = \phi\left(z_i^{(l-1)}(x)\right) \tag{1}$$

where:

$$z_i^{(l)}(x) = b_i^{(l)} + \sum_{j=1}^{N_{l-1}} W_{ij}^{(l)} h_j^{(l-1)}(x). \tag{2}$$

For convenience, the parameters of the neural network are combined into a vector of parameters $\theta = \left(b^{(1)}, W^{(1)}, \ldots, b^{(L)}, W^{(L)}\right)$ and input data $h^{(0)} = x = (1, x_1, \ldots, x_{N_l})$. Deep neural networks compose computations performed by many layers. The calculation for a network with L hidden layers is:

$$\hat{y} = f\left[h^{(L)}\left(z^{(L)}\left(h^{(L-1)}\left(\ldots z^{(2)}\left(h^{(2)}\left(z^{(1)}\left(h^{(1)}\left(z^{(0)}\right)\right)\right)\right)\right)\right)\right)\right]. \tag{3}$$

When L is large, it is called a deep neural network, and each pre-activation function $z^{(L)}(x)$ is typically a linear operation with the matrix $W^{(L)}$ and bias $b^{(L)}$, which can be combined with the parameters θ. An architecture of a feedforward neural network with L hidden layers is shown in Figure 1.

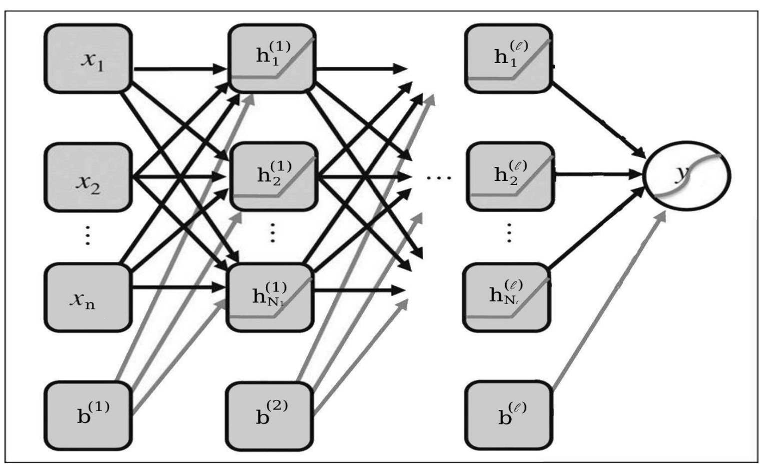

Figure 1: An architecture of a feedforward neural network with l hidden layers, (Witten et al., 2016) [29].

3 Gaussian processes

Assume we have access to a training dataset of n input-output observations

$$\mathbb{D} = \{(x_i, y_i) : i = 1, \dots, n\} \tag{4}$$

y_i is assumed to be a noisy realisation of an underlying latent function $f = f(x)$, that is,

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma_\epsilon^2) \tag{5}$$

where $x_i \in \mathbb{R}^q$ and $y_i \in \mathbb{R}$.

The interest here is in estimating the function f, which generally is non-linear. The GP provides a natural way to make inferences about these functions (Rasmussen and Williams, (2006), [23]). The GP prior over functions can be seen as an extension of the multivariate Gaussian distribution. By definition, a stochastic process is a set of random variables $\{f(x) : x \in \mathcal{X}\}$, indexed by a set, $\mathcal{X}$. A GP is a stochastic process such that for any finite set of function evaluations, $f(x) = (f(x_1), \dots, f(x_n))^T$, where f is multivariate Gaussian distributed. We say that for any finite set of elements drawn from $\mathcal{X}$, f is a GP described by a mean, $m(.)$, and covariance function, $K(.,.)$, which we write as:

$$f(x) \sim GP\left(m(x), K\left(x, x'\right)\right) \tag{6}$$

where

$$m(x) = \mathbb{E}\left[f(x)\right], \quad K\left(x_i, x_j\right) = \mathbb{E}\left\{\left[f(x_i) - m(x_i)\right]\left[f(x_j) - m(x_j)\right]\right\} \tag{7}$$

$$m : \mathcal{X} \to \mathbb{R}, \quad \text{and} \quad K(.,.) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}. \tag{8}$$

Let X_{new} be a matrix with on each row a new input point x_i^{new}, $i = 1, \dots, n$. To sample a function, we first compute the covariances between all inputs in $X_{new} = (x_1^{new}, \dots, x_n^{new})$ and collect these in an $n \times n$ matrix:

$$K\left(X_{new}, X_{new}\right) = \begin{pmatrix} k\left(x_1^{new}, x_1^{new}\right) & \dots & k\left(x_1^{new}, x_n^{new}\right) \\ k\left(x_2^{new}, x_1^{new}\right) & \dots & k\left(x_2^{new}, (x_n^{new}\right) \\ \vdots & \vdots & \vdots \\ k\left(x_n^{new}, x_1^{new}\right) & \dots & k\left(x_n^{new}, x_n^{new}\right) \end{pmatrix}.$$

Let

$$f_{new} = \left(f\left(x_1^{new}\right), \dots, f\left(x_n^{new}\right)\right)^T.$$

The joint distribution$p(f, f_{new})$ is given by:

$$\begin{pmatrix} f \\ f_{new} \end{pmatrix} \sim N\left[\begin{pmatrix} m(X) \\ m(X_{new}) \end{pmatrix}, \begin{pmatrix} K(X,X) & K(X,X_{new}) \\ K(X_{new},X) & K(X_{new},X_{new}) \end{pmatrix}\right]$$

where $K(X, X)$, represents the kernel evaluated at X, $K(X_{new}, X_{new})$ is the covariance matrix between the new points, $K(X, X_{new})$ is the covariance matrix between the observed points and the new values, and $K(X_{new}, X)$ is the covariance matrix between the new and the observed points.
The covariance function must be positive definite, that is

$$\sum_j \sum_i v_i K(x_i, x_j) v_j \geq 0, \quad \text{for all} \quad v_i,\ x_i. \tag{9}$$

To complete the prior specification, we need to specify the mean and covariance functions. Any positive definite covariance function can be chosen; for $K(.,.)$ are the following, see Table 1.

Table 1: Covariance functions.

Kernel	$K(x, x')$
Exponential	$\sigma^2 \exp\left(-\frac{\lvert x-x'\rvert}{l}\right)$
Squared Exponential	$\sigma^2 \exp\left(-\frac{1}{2}\frac{\lvert x-x'\rvert^2}{l^2}\right)$
Matérn	$\sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\lvert x-x'\rvert}{l}\right)^{\nu} \kappa_\nu\left(\frac{\lvert x-x'\rvert}{l}\right)$
Brownian Motion	$\min(x, x')$

The distribution predicted by the GP can be determined by the conditional rules of multivariate Gaussian distribution:

$$f_{new}|f, X, y \sim N\left(\mathbb{E}\left(f_{new}\right), \mathbb{C}ov\left(f_{new}\right)\right) \tag{10}$$

where

$$\mathbb{E}\left(f_{new}\right) = m\left(X_{new}\right) + K\left(X_{new}, X\right) K^{-1}\left(X, X\right)\left[f - m\left(X\right)\right] \tag{11}$$

and

$$\mathbb{C}ov\left(f_{new}\right) = K\left(X_{new}, X_{new}\right) - K\left(X_{new}, X\right) K^{-1}\left(X, X\right) K\left(X, X_{new}\right).$$

For a Gaussian likelihood:

$$y_i = f(x_i) + \epsilon_i \quad , \quad \epsilon_i \sim N\left(0, \sigma_\epsilon^2\right) \quad , \quad y|f \sim N\left(f(x_i), \sigma_\epsilon^2 I\right) \tag{12}$$

where I is the identity matrix. The noise can be included in the covariance function, as follows:

$$K\left(f(x_i), f(x_j)\right) = K\left(x_i, x_j\right) + \delta_{ij}\sigma_\epsilon^2 \tag{13}$$

where δ_{ij} is the Kronecker delta, and σ_ϵ^2 is the noise variance between layers.

The uncertainty is now present in the observations, and the joint distribution over the unknown data and the known data is augmented in the covariance equation by

$$\begin{pmatrix} f \\ f_{new} \end{pmatrix} \sim N\left(\begin{pmatrix} m\left(X\right) \\ m(X_{new}) \end{pmatrix}; \begin{pmatrix} K\left(X,X\right) + \sigma_\epsilon^2 I & K\left(X, X_{new}\right) \\ K\left(X_{new}, X\right) & K\left(X_{new}, X_{new}\right) \end{pmatrix}\right).$$

The marginal distribution is given by

$$f_{new}|X_{new}, X, f \sim N\left(\mathbb{E}\left(y_{new}\right), \mathbb{C}ov\left(y_{new}\right)\right) \tag{14}$$

where

$$\mathbb{E}\left(y_{new}\right) = m\left(X_{new}\right) + K\left(X_{new}, X\right)\left(K\left(X,X\right) + \sigma_\epsilon^2 I\right)^{-1}\left[y - m\left(X\right)\right]$$

and

$$\mathbb{C}ov\left(y_{new}\right) = K\left(X_{new}, X_{new}\right) - K\left(X_{new}, X\right)\left(K\left(X,X\right) + \sigma_\epsilon^2 I\right)^{-1} K\left(X, X_{new}\right).$$

In this case the integrals required to infer a posterior, $p\left(f_{new}|X_{new}, X, f\right)$, are tractable.

4 Deep neural networks and Gaussian processes

In this article, we consider fully-connected ANNs with layers numbered from $l = 0$ (input) to $l = L - 1$ (output), each containing, $N_0, \ldots, N_{L-1}$ neurons, and with a Lipschitz, twice differentiable nonlinearity activation function $\phi : \mathbb{R} \to \mathbb{R}$, with bounded second derivative. For each $x \in \mathbb{R}^{d_{in}}$ denote the input to the network ($x = (x_1, \ldots, x_{d_{in}})$), and $z^{(l)} \in \mathbb{R}^{d_{out}}$ denote its output. We use $z_i^{(l)}(x), h_i^{(l)}(x)$ to represent the pre- and post-activation functions at layer l with input x, also, let $h_i^{(0)} = x$. The parameters consist of the connection matrices $W_{ij}^{(l)} \in \mathbb{R}^{N_l \times N_{l+1}}$ and bias vectors $b_i^{(l)} \in \mathbb{R}^{N_{l+1}}$ for $l = 0, \ldots, L-1$; which are independent and randomly selected, with zero mean and variances $\frac{\sigma_w^2}{N_l}$, and σ_b^2, respectively.

Now we are going to establish the relationship between a single-hidden layer neural networks, and Gaussian processes. Suppose that $z_j^{(l)}(x)$ is a Gaussian process with mean and covariance functions $\mu^{(l)}(x)$, $K^{(l)}(x, x')$, respectively, that is,

$$z_j^{(l)}(x) = \begin{pmatrix} z_j^{(l)}(x) \\ z_j^{(l)}(x') \end{pmatrix} \sim GP\left(\mu^{(l)}(x), K^{(l)}\left(x, x'\right)\right)$$

where

$$\mu^{(l)}(x) = \mathbb{E}\left\{\mathbf{z}_j^{(l)}(x)\right\} = \begin{pmatrix} \mu(x) \\ \mu(x') \end{pmatrix}$$

and

$$K = K^{(l)}\left(x, x'\right) = \mathbb{C}ov\left\{z_j^{(l)}(x), z_j^{(l)}(x')\right\} = \begin{pmatrix} K^{(l-1)}(x,x) & K^{(l-1)}(x,x') \\ K^{(l-1)}(x',x) & K^{(l-1)}(x',x') \end{pmatrix}.$$

The $i-$th component of the network output, $z_i^{(1)}$, is computed as:

$$z_i^{(1)}(x) = b_i^{(1)} + \sum_{j=1}^{N_1} W_{ij}^{(1)} h_j^{(1)}(x),\ h_j^{(1)}(x) = \phi\left(b_j^{(0)} + \sum_{k=1}^{d_{in}} W_{ij}^{(0)} x_k\right). \quad (15)$$

Note that there is a dependency on the input data vector x, and also since the weight and bias are considered i.i.d., the post-activations $h_j^{(1)}$ and $h_k^{(1)}$ are independent for $j \neq k$. Also, $z_i^{(1)}(x)$ is a sum of i.i.d terms, it follows from the Central Limit Theorem that in the limit of infinite width $N_1 \to \infty$, $z_i^{(1)}(x)$ will be Gaussian distributed, Lee, et. al (2018) [14].

Suppose that $z_j^{(l)}(x)$ is a Gaussian processes, i.i.d for every j, and that $h_j^{(l)}(x)$, also are independent and identically distributed. Then after $l-1$ steps, the recurrence relation for a feedforward network is defined as

$$z_i^{(l)}(x) = b_i^{(l)} + \sum_{j=1}^{N_l} W_{ij}^{(l)} h_j^{(l)}(x), \quad h_j^{(l)}(x) = \phi\left(z_j^{(l-1)}(x)\right). \quad (16)$$

Prior on weights:

$$b_j^{(l)}|\sigma_b^2 \sim N\left(0, \sigma_b^2\right), \quad W_{ij}^{(l)}|\Sigma_W \sim N\left(0, \Sigma_W\right). \quad (17)$$

Prior on hyperparameters:

$$\sigma_b^2|\alpha, \beta \sim IG\left(\alpha, \beta\right) \quad \Sigma_W|\nu, R \sim IG\left(\nu, R\right). \quad (18)$$

Note, $z_i^{(l)}(x)$ is a sum of i.i.d. random terms so that, as $N_l \to \infty$, any finite collection $\{z_i^{(l)}(x), \quad l = 1, \dots, L\}$ will have joint multivariate Gaussian distribution, i.e,

$$\mathbf{z}_i^{(l)}(x) \sim GP(\mu^{(l)}(x), K^{(l)}\left(x, x'\right)),\ \mathbf{z}_i^{(l)}(x) = \left(z_i^{(l)}(x_1), \dots, z_i^{(l)}(x_n)\right). \tag{19}$$

A general equation is now established to approximate the covariance for a bivariate Gaussian process:

$$\begin{aligned}
K^{(l)}\left(x, x'\right) &= \mathbb{E}\left\{\left[z_i^{(l)}(x) - \mathbb{E}\left(z_i^{(l)}(x)\right)\right]\left[z_i^{(l)}(x') - \mathbb{E}\left(z_i^{(l)}(x')\right)\right]\right\} \\
&= \mathbb{E}\left\{\left[b_i^{(l)} + \sum_{j=1}^{N_l} W_{ij}^{(l)} h_j^{(l)}(x)\right]\left[b_i^{(l)} + \sum_{j=1}^{N_l} W_{ij}^{(l)} h_j^{(l)}(x')\right]\right\} \\
&= \sigma_b^2 + \Sigma_W \sum_{j=1}^{N_l} \mathbb{E}\left[h_j^{(l)}(x) h_j^{(l)}(x')\right] \\
&= \sigma_b^2 + \Sigma_W \sum_{j=1}^{N_l} \mathbb{E}\left[\phi\left(z_j^{(l-1)}(x)\right)\phi\left(z_j^{(l-1)}(x')\right)\right]
\end{aligned} \tag{20}$$

where the calculation of the expectation is a two dimensions Gaussian integral:

$$\begin{aligned}
\mathbb{E}\left(\phi\left(z_j^{(l-1)}(x)\right)\phi\left(z_j^{(l-1)}(x')\right)\right) = &\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \phi\left(z_j^{(l-1)}(x)\right)\phi\left(z_j^{(l-1)}(x')\right) \\
&\times p\left(z_j^{(l-1)}(x), z_j^{(l-1)}(x')\right) \\
&\times dz_j^{(l-1)}(x) dz_j^{(l-1)}(x').
\end{aligned} \tag{21}$$

Since:

$$\begin{pmatrix} z_j^{(l)}(x) \\ z_j^{(l)}(x') \end{pmatrix} \sim GP\left(\mu^{(l)}(x), K^l\left(x, x'\right)\right) \tag{22}$$

then

$$\begin{aligned}
&\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \phi\left(z_j^{(l-1)}(x)\right)\phi\left(z_j^{(l-1)}(x')\right)\frac{1}{2\pi|K|^{\frac{1}{2}}} \\
&\times \exp\left\{-\frac{1}{2}\Big(z_j(x)^{(l-1)} - \mu(x), \quad z_j^{(l-1)}(x') - \mu(x')\Big) K^{-1}\begin{pmatrix} z_j(x)^{(l-1)} - \mu(x) \\ z_j^{(l-1)}(x') - \mu(x') \end{pmatrix}\right\} \\
&\times dz_j^{(l-1)}(x) dz_j^{(l-1)}(x').
\end{aligned}$$

Let

$$K^{-1} = \left(\sqrt{K}^{T}\right)^{-1}\left(\sqrt{K}\right)^{-1}. \tag{23}$$

Consider the following transformation:

$$\begin{pmatrix}\xi_{i_1}\\ \xi_{i_2}\end{pmatrix} = \left(\sqrt{K}\right)^{-1}\begin{pmatrix} z_j(x)-\mu(x)\\ z_j(x')-\mu(x')\end{pmatrix} \Rightarrow \left(\sqrt{K}\right)\begin{pmatrix}\xi_{i_1}\\ \xi_{i_2}\end{pmatrix} = \begin{pmatrix} z_j(x)-\mu(x)\\ z_j(x')-\mu(x')\end{pmatrix}.$$

To simplify the notation, the following variable change is made:

$$z_j(x) = z_j(x)^{(l-1)}, \quad z_j(x') = z_j(x')^{(l-1)}. \tag{24}$$

$$\begin{aligned}\sqrt{K}\xi_{i_1} = z_j(x)-\mu(x) &\Rightarrow z_j(x) = \sqrt{K}\xi_{i_1}+\mu(x)\\ &\iff \xi_{i_1} = \frac{z_j(x)-\mu(x)}{\sqrt{K}} \sim N(0,1)\end{aligned}$$

and

$$\begin{aligned}\sqrt{K}\xi_{i_2} = z_j(x')-\mu(x') &\Rightarrow z_j(x') = \sqrt{K}v+\mu(x')\\ &\iff \xi_{i_2} = \frac{z_j(x')-\mu(x')}{\sqrt{K}} \sim N(0,1).\end{aligned}$$

The Jacobian of the transformation is:

$$J = \begin{pmatrix}\frac{\partial z_j(x)}{\partial \xi_{i_1}} & \frac{\partial z_j(x)}{\partial \xi_{i_2}}\\ \frac{\partial z_j(x')}{\partial \xi_{i_2}} & \frac{\partial z_j(x')}{\partial \xi_{i_2}}\end{pmatrix} = \begin{pmatrix}\sqrt{K} & 0\\ 0 & \sqrt{K}\end{pmatrix} \Rightarrow |J| = |K|^{\frac{1}{2}} \tag{25}$$

then

$$\begin{aligned}&\frac{1}{2\pi|K|^{\frac{1}{2}}}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\phi\left(\sqrt{K}\xi_{i_1}+\mu(x)\right)\phi\left(\sqrt{K}\xi_{i_2}+\mu(x')\right)\\ &\times\frac{1}{2\pi}\exp\left(-\frac{1}{2}\left(\xi_{i_1}^2+\xi_{i_2}^2\right)\right)|J|d\xi_{i_1}d\xi_{i_2}\\ &=\frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\phi\left(\sqrt{K}\xi_{i_1}+\mu(x)\right)\phi\left(\sqrt{K}\xi_{i_2}+\mu(x')\right)\\ &\times\exp\left(-\frac{1}{2}\xi_{i_1}^2\right)\exp\left(-\frac{1}{2}\xi_{i_2}^2\right)d\xi_{i_1}d\xi_{i_2}\\ &=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\phi\left(\sqrt{K}\xi_{i_1}+\mu(x)\right)\exp\left(-\frac{1}{2}\xi_{i_1}^2\right)d\xi_{i_1}\\ &\times\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\phi\left(\sqrt{K}\xi_{i_2}+\mu(x')\right)\exp\left(-\frac{1}{2}\xi_{i_2}^2\right)d\xi_{i_2}.\end{aligned}$$

The approximation of the last integral depends on the choice of the sigmoid function $\phi(.)$.

In the case of deep neural networks, the transfer function $\phi(z)$ is a bounded function where all moments are bounded. Then we can apply Central Limit Theorem to show that the stochastic process is a Gaussian process, Williams (1996) [27] and Lee et. al. (2018) [14].

Cho and Saul (2009) [5] developed a new family of covariance functions which allows computing the correlation between two vectors $x, z \in \mathbb{R}^{N_l}$. They define the $n-$th order arc-cosine kernel function via the integral representation:

$$K^{(l)}(x,z)=2\int \frac{1}{(2\pi)^{\frac{N_l}{2}}} \exp\left(-\frac{\|w\|^2}{2}\right)\Theta(w\,.x)\Theta\left(w\,.z\right)\left(w\,.x\right)^l\left(w\,.z\right)^l dw, \tag{26}$$

where $\Theta\left(t\right) = \frac{1}{2}\left(1 + sign(t)\right)$ denote the Heaviside step function. The integral representation (26) allows the kernel of covariance functions to be positive definite, and that the dependence between x y z, can be written as:

$$K^{(l)}\left(x,z\right) = \frac{1}{\pi}\|x\|^l\|z\|^l J_l(\theta) \tag{27}$$

where all the angular dependence is captured by $J_l(\theta)$. The angular dependence is given by:

$$J_l(\theta) = (-1)^l\left(\sin\theta\right)^{2l+1}\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\right)^l\left(\frac{\pi-\theta}{\sin\theta}\right). \tag{28}$$

In particular, when $l = 0$, we have the angle between x and z. When $l > 0$, the angular dependence is more complicated. Some terms of $J_l(\theta)$, are shown:

$$J^0(0) = \theta, \quad J^1(\theta) = \sin\theta + (\pi - \theta)\cos\theta,$$
$$J^2(\theta) = 3\sin\theta\cos\theta + (\pi-\theta)\left(1 + 2\cos^2\theta\right).$$

The arc-cosine kernel for $l = 0$, is represented by:

$$K^{(0)} = 1 - \arccos^{-1}\left(\frac{xz}{\|x\|\|z\|}\right). \tag{29}$$

Neural network models are strongly related to the kernel of functions defined in (26), when considering the inner product between the different outputs of the neural network as:

$$\phi(x)\phi(z) = \sum_{i=1}^{m}\Theta(w_i\,.x)\Theta\left(w_i\,.z\right)\left(w_i\,.x\right)^l\left(w_i\,.z\right)^l \tag{30}$$

where w_i denote $i-$th row of the weight matrix W and m is the number of output units. In the limit, it can be seen that the equation (30) is equivalent to (26), Cho and Saul (2009), [5]:

$$\lim_{m\to\infty}\frac{2}{m}\phi(x)\phi(z) = K^{(l)}(x,z). \tag{31}$$

Also, Cho and Saul (2009) [5] proved that for a vector of inputs $x = (1, x_1, \ldots, x_{N_l})$, their characteristics can be mapped by means of a non-linear transformation $\phi(x)$, using kernel functions:

$$K^{(l)}(x,z) = \phi\left(\phi\left(\ldots\phi(x)\right).\phi\left(\ldots\phi(z)\right)\right). \tag{32}$$

The iterated equation (32) mimics a multilayer neural network, for example for a one-layer neural network, $K(x,z) = \phi(x)\phi(z)$. Cho and Saul (2009) [5] define a recursive kernel through a new mapping of features through compositions such as $\phi(\phi(x))$. In the case of a linear kernel $K(x,z) = xz$, the composition is $\phi(\phi(x)) = \phi(x) = x$, and for homogeneous polynomial kernels $K(x,z) = (xz)^{N_l}$; the composition is:

$$K(x,z) = \phi\left(\phi(x)\right).\phi\left(\phi(z)\right) = \left(\phi(x)\phi(z)\right)^{N_l} = \left((x.z)^{N_l}\right)^{N_l} = (x.z)^{N_l^2}. \tag{33}$$

Then we consider the composition of l layers based on the iterate defined in (32), applying a mathematical induction the inductive step is given by:

$$K^{(l)}(x,z) = \frac{1}{\pi}\left[K^{(l-1)}(x,x)K^{(l-1)}(z,z)\right]^{\frac{l}{2}} J_l\left(\theta^{(l)}\right) \tag{34}$$

where $\theta^{(l)}$ is the angle between x and y in the feature space:

$$\theta^{(l)} = \cos^{-1}\left\{\frac{K^{(l)}(x,z)}{\left[K^{(l)}(x,x)K^{(l)}(z,z)\right]^{\frac{1}{2}}}\right\}. \tag{35}$$

Recently, a linear rectified function $ReLU(t) = \max(0;t)$, was successfully used in neural networks as it carries the neuron signal better. ReLUs have the desirable property that they do not require input normalization, Krizhevsky et al. (2012) [12]. To compute the given integral (21), a rectified linear sigmoidal can be used, Hazan et al. (2015) [9], which results in an analytical kernel given by

$$K_{ReLU}(x_i,x_j) = \frac{\|x_i\|\|x_j\|}{\pi}\sin\left(cos^{-1}(\rho_{ij})\right) + \left(\pi - cos^{-1}(\rho_{ij})\right)\rho_{ij}$$

where

$$\rho_{ij} = \frac{\langle x_i, x_j \rangle}{\|x_i\| \|x_j\|}, \quad \langle x_i, x_j \rangle = \int \int \phi(x_i)\phi(x_j) p\left(x_i, x_j\right) dx_i dx_j.$$

To compute the entries of $K(.,.)$, let $\phi(t) = ReLU(t)$,

$$\begin{pmatrix} z_1' \\ z_2' \end{pmatrix} \sim N\left(0, K\right), \quad 0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad K = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$$

then

$$\begin{aligned} \mathbb{E}_{z_1', z_2'}\left(\phi\left(z_1'\right)\phi\left(z_2'\right)\right) &= h\left(\sigma_1, \sigma_2, \rho\right) \\ &= \frac{\sigma_1\sigma_2}{\pi} \sin\left(\cos^{-1}(\rho)\right) + \rho\left(\pi - \cos^{-1}(\rho)\right) \end{aligned}$$

and $K_{ReLU}^{(1)}$ is given by

$$\begin{pmatrix} h\left(\sqrt{1+\alpha}\|x_i\|, \sqrt{1+\alpha}\|x_i\|, \frac{\alpha}{1+\alpha}\right) & h\left(\sqrt{1+\alpha}\|x_i\|, \sqrt{1+\alpha}\|x_j\|, \frac{\alpha}{1+\alpha}\frac{\langle x_i, x_j\rangle}{\|x_i\|\|x_j\|}\right) \\ h\left(\sqrt{1+\alpha}\|x_i\|, \sqrt{1+\alpha}\|x_j\|, \frac{\alpha}{1+\alpha}\frac{\langle x_i, x_j\rangle}{\|x_i\|\|x_j\|}\right) & h\left(\sqrt{1+\alpha}\|x_j\|, \sqrt{1+\alpha}\|x_j\|, \frac{\alpha}{1+\alpha}\right) \end{pmatrix}.$$

Iterating successively, we obtain:

$$K_{ReLU}^{(2)}\left(x_i, x_j\right) = \mathbb{E}_{z_1, z_2}\left(\phi\left(z_1\right)\phi\left(z_2\right)\right), \quad \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \sim N\left(0, K\right)$$

is a recursive equation of h with the appropriate parameters, see Hazan et al. (2015) [9].

On the other hand, using GP prior functions allows Bayesian inference to be made precisely to obtain predictions and estimate the uncertainty in deep neural network models. The estimation of the parameters is not required through training based on a gradient-type algorithm.

Under the Bayesian statistical approach, the weights and biases of the network are generated following a probability distribution $p(\mathbf{W}|\theta)$, where ($\mathbf{W} = (W, b)$, represents the weights and bias, and $\theta = (\alpha, \beta, \nu, R) \sim p(\theta)$ represents the hyperparameters, which can be integrated.

$$p\left(\mathbf{W}\right) = \int p(\mathbf{W}|\theta)p(\theta)d\theta. \tag{36}$$

Given a training sample $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $\mathbf{x} = (x_1, \ldots, x_n)^T$ and $\mathbf{y} = (y_1, \ldots, y_n)^T$ denote the input and output of the network, respectively. Suppose now that we want to make a prediction with a test data x_{new} using priors over funtions rather than weights $\mathbf{z}(x) = (z_1, \ldots, z_n)$ restricted to input values $\mathbf{x}$. Then

$$
\begin{aligned}
p\left(z_{new}|\mathcal{D}, x_{new}\right) &= \int p\left(z_{new}|\mathbf{z}, \mathbf{x}, x_{new}\right) p\left(\mathbf{z}|\mathcal{D}\right) d\mathbf{z} \\
&= \frac{1}{p(\mathbf{y})} \int p\left(z_{new}, \mathbf{z}|x_{new}, \mathbf{x}\right) p\left(\mathbf{y}|\mathbf{z}\right) d\mathbf{z} \qquad (37)
\end{aligned}
$$

where $\mathbf{y} = (y_1, \ldots, y_n)^T$ are the targets on the training set, $p(\mathbf{y})$ is a marginal likelihood, and $p\left(\mathbf{y}|\mathbf{z}\right)$ corresponds to observation noise. We will assume a noise consisting of a Gaussian $(y|z \sim N(0, \sigma_\epsilon^2))$.

The importance of choosing priors over functions implies that $z_1, \ldots, z_n$, z_{new} are generated from a Gaussian processes

$$
\begin{pmatrix} z \\ z_{new} \end{pmatrix} \sim GP\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} K\left(\mathcal{D}, \mathcal{D}\right) & K\left(x_{new}, \mathcal{D}\right) \\ K\left(\mathcal{D}, x_{new}\right) & K\left(x_{new}, x_{new}\right) \end{pmatrix}\right].
$$

Then the integral in (37) can be obtained exactly, since the marginal distribution is:

$$
z_{new}|\mathcal{D}, x_{new} \sim N\left(\mu_{post}, K_{post}\right) \qquad (38)
$$

where

$$
\mu_{post} = K\left(x_{new}, \mathcal{D}\right) \left(K\left(\mathcal{D}, \mathcal{D}\right) + \sigma_\epsilon^2 I_n\right)^{-1} y
$$

and

$$
K_{post} = K\left(x_{new}, x_{new}\right) - K\left(x_{new}, \mathcal{D}\right) \left(K\left(\mathcal{D}, \mathcal{D}\right) + \sigma_\epsilon^2 I_n\right)^{-1} K^T(x_{new}, \mathcal{D})
$$

where I_n is the $n \times n$ matrix identity. The predicted distribution of $z_{new}|\mathcal{D}$, $\mathbf{x}$ is clearly determined. Deep neural network training works using a Bayesian approach. The covariance function is determined by choosing a prior Gaussian process. In this case, the model depends on the depth, non-linearity of the transfer function, weights and biases.

Finally, the results obtained through the estimation method by a neural network approximated by a Gaussian process are compared with those obtained by a neural network. The weights are optimized using Adam's variant of stochastic gradient descent (SGD) algorithm, using a loss function MSE, Lee et. al. (2018) [14].

5 Experimental analysis

The experimental goal is to validate the performance of the proposed method in diagnosing Alzheimer's disease. Different experiments were executed, varying the size of the input set, to view the accuracy difference when the input size increased. As a reference point, this section compares the results of the proposed method with multi-layer neuronal networks.

5.1 Data description

The data set was used by a previous investigation of Alzheimer's disease (AD) detection. AD is a progressive and irreversible brain disorder that causes memory problems, slowly destroying it until losing the ability to develop simple tasks. A non-invasive method to study AD uses Electroencephalograms (EEGs) that register the brain's electrical activity. However, the EEG raw signals are challenging to classify directly. Martínez et . (2021) proposed to extract five significant features provided by public EGG experiments from `https://osf.io/jbysn/`. In addition, for this study, we only considered the five critical features mentioned in Table 2.

Table 2: Features extracted from the EEGs to identify the AD effects.

Notation	Feature Name
HFD	Higuchi Fractal Dimension
LZC	Lempel Ziv Complexity
PSD	Power Spectral Density
RP	Relative Power
SE	Sample Entropy

Each data set has 16 dimensions corresponding to each channel in an EGG test registration. However, for *Relative Power* feature (plots in section 5.3), the input size is 64. Also, there is an extra column for the label class in all files.

In this experiment, we cleaned up the data of significant features by randomly removing data to leave the same number of labels for each class in the data set. In total, 514 observations were considered, of which 85% were for training data and the rest 15% for testing.

5.2 Experiment description

The main goal is to use the above data set to classify the positive and negative cases of Alzheimer's disease. We use the classification models in Table 3.

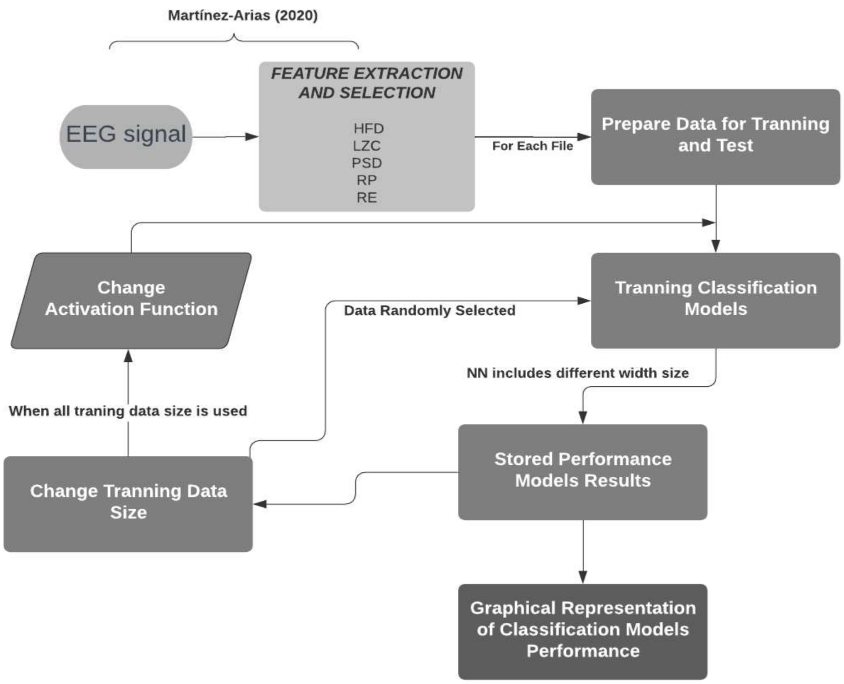

Figure 2: Experiment description diagram.

Each file was used to train the GP and NN models independently. That is, in total, results are obtained from 7 classification models (GP, $NN = 5$, $NN = 10$, $NN = 50$, $NN = 100$, $NN = 500$, and $NN = 1000$). Comparing the performance of GPs against neural networks changes parameters such as network width, data set size and activation function. In each model, the training data size for the learning process was changed; that is, $n = 75$, $n = 100$, $n = 250$ and $n = 436$ observations are taken to train the model for each of the data-set sizes. Similarly, all models are tested with the ReLu and Tanh activation functions. Also, in neural networks, the width of the network is changed to compare its performance against the Gaussian process. In Figure 2 there is a diagram that explains the experiment for each data set.

Lastly, in the following repository there is a Python code implementation of this work, and the data sets used, `https://github.com/KrishnaRoman/Deep-Gaussian-Processes`

Table 3: Notation of the classification models used in this work.

Notation	Model Name
NN-5	Neural Network with width 5
NN-10	Neural Network with width 10
NN-50	Neural Network with width 50
NN-100	Neural Network with width 100
NN-500	Neural Network with width 500
NN-1000	Neural Network with width 1000

5.3 Performance results

The results for the Relative Power feature are shown in Figure 3. Clearly, the model that uses the proposed Gaussian Process has the best test accuracy. This figure shows that the Neural Network models behave better with Tanh activation function than Relu on this feature. While in the case of the GP there is no significant difference. Additionally, the performance results on the other features are shown in Table 4.

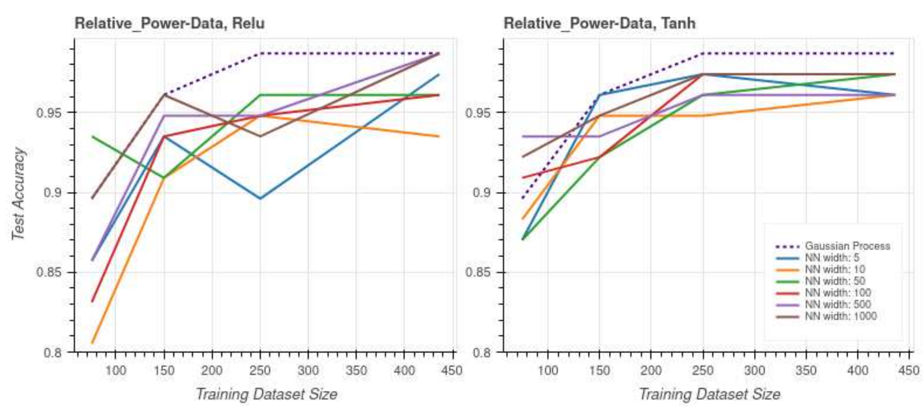

Figure 3: Relative power performance results.

5.4 Error results

The predictive performance of Deep Gaussian Process (DGP) and Neural Networks (NN) models are evaluated using the following metric, Mean Squared Error (MSE), defined by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{39}$$

where $\hat{y}_i$ indicates the predicted value, y_i indicates the actual values, and n is the number of predictions. The metric range is in $[0, \infty)$ and lower values indicate better performance. It can be seen in Figure 4 for Relative Power that in terms of error, the GP with the Relu activation function is the best, whereas the Neural Network with 5-width is better in Tanh. Also, the error results on the other features are shown in Table 4.

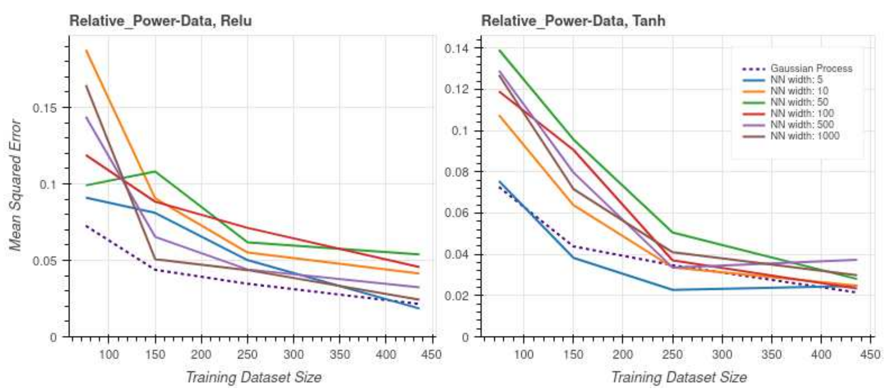

Figure 4: Relative power error results.

The metrics for all features and models were recorded in Table 4. In this case only was considered the results with the highest dataset size that is 436.

6 Discussion and conclusions

Using a nonstochastic gradient-based training, we use EEG Signals for diagnosing Alzheimer's Disease to test our GP's behavior, which uses a hierarchical Bayesian structure to model the weights and biases of a neural network, and compare it with neural networks varying their widths. A general formula was derived to evaluate the resulting integrals of Gaussian processes with non-linear

Table 4: Performance and error results for all features and models.

Dataset	Model (ReLU)	Test accuracy	Test MSE	Model (tanh)	Test accuracy	Test MSE
LZC	GP	0.935	0.0564	GP	0.948	0.1092
	NN-10	0.935	0.0508	NN-10	0.883	0.0784
	NN-1000	0.961	0.0404	NN-1000	0.948	0.059
RP	GP	0.987	0.0215	GP	0.974	0.0221
	NN-10	0.935	0.0414	NN-10	0.961	0.0248
	NN-1000	0.987	0.0243	NN-1000	0.974	0.0299
HFD	GP	1.000	0.0125	GP	0.987	0.0153
	NN-10	0.961	0.0243	NN-10	1.000	0.0152
	NN-1000	1.000	0.0114	NN-1000	1.000	0.0178
PSD	GP	0.974	0.0246	GP	0.987	0.0245
	NN-10	0.961	0.0244	NN-10	0.961	0.0215
	NN-1000	0.987	0.0157	NN-1000	0.974	0.0207
SE	GP	1.000	0.0125	GP	0.987	0.0153
	NN-10	0.961	0.0243	NN-10	1.000	0.0152
	NN-1000	1.000	0.0114	NN-1000	1.000	0.0178

transfer functions, and obtained a kernel to update the covariance functions. The proposed methodology was applied to the classification of EEG signals for diagnosing Alzheimer's disease, considering five data sets and estimating the models varying size of the samples. This study showed that DGP can be used in supervised learning and classification tasks. As mentioned in Lee et al. (2018), we check that the GP behaves as a neural network with an infinite number of neurons in the hidden layers. In general, all test accuracies were very high because the data set consisted on a limited number of patients and the work done by Martínez-Arias (2020) [17] on feature extraction and selection.

In our experiments, we realize that inverse gamma distribution hyper-parameters fixed well for the model learning process. The bias and weight variances are randomly chosen using these hyper-parameters, and it contributes to evaluating the model performance. The results show that GP classifies well and sometimes even better than neural network models. In this sense, we proved that Deep Gaussian Model working with regression data is a Gaussian Process when the neural network width tends to be infinite. Also, this model is more effective when the data size is more extensive. Another big difference between DGP against traditional neural network models is the use of GP properties like precise uncertainty estimates.

For future research, we expect to prove our DGP model to classify other data types, such as crude data or images. Crude ECG signal data could be an interesting scenario to use DGP model to organize positive cases of Alzheimer's disease. Also, analyze what role distribution hyper-parameters play in the model with other data types.Finally, the integral I_z computed in section 4 may be used to estimate the kernel using a different method that we proposed. This could be another research to see which alternative method works better and to compare the GP performance results.

Acknowledgements

We want to express our special thanks to the journal's anonymous reviewers for their suggestions to improve the manuscript. In the same way, we thank the authors of the previous works who provided us with the data for our research.

Funding

We also thank the project Functional Data Analysis: Methods and Applications, with registration REG-INV-19-04054, Yachay Tech.

References

[1] D. Abásolo, J. Escudero, R. Hornero, C. Gómez, P. Espino, *Approximate entropy and auto mutual information analysis of the electroencephalogram in Alzheimer's disease patients*, Medical & Biological Engineering & Computing **46** (2008), 1019–1028. Doi: 10.1007/s11517-008-0392-1

[2] D. Agrawal, T. Papamarkou, J. Hinkle, *Wide neural networks with bottlenecks are deep Gaussian processes*, Journal of Machine Learning Research **21** (2020), no. 175, 1–66. Available from: Link

[3] N. Cedeño, G. Carillo, M.J. Ayala, S. Lalvay, S. Infante, *Analysis of chaos and predicting the price of crude oil in Ecuador using deep learning models*, in: T. Guarda, F. Portela & M.F. Santos (Eds.) Advanced Research in Technologies, Information, Innovation and Sustainability, part of Communications in Computer and Information Science series **1485**, Springer, Cham, 2021, 318–332. Doi: 10.1007/978-3-030-90241-4_25

[4] N. Cedeño, S. Infante, *Estimation of ordinary differential equations solutions with Gaussian processes and polynomial chaos expansion*, in: J.P. Salgado Guerrero, J. Chicaiza Espinosa, M. Cerrada Lozada, S. Berrezueta-Guzman (Eds.) Information and Communication Technologies, part of Communications in Computer and Information Science series **1456**, Springer, Cham, 2021, pp. 3–17. Doi: : 10.1007/978-3-030-89941-7_1

[5] Y. Cho, L. Saul, L. *Kernel methods for deep learning*, in: Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams & A. Culotta (Eds.) Advances in Neural Information Processing Systems **22** (2009), 342–350. Available from: Link

[6] A. Damianou, N. Lawrence, *Deep Gaussian processes*, Proceedings of the 16th International Conference on Artificial Intelligence and Statistics, Scottsdale AZ, U.S.A. (2013), 207–215. Available from: Link

[7] A. Garriga-Alonso, C.E. Rasmussen, L. Aitchison, *Deep convolutional networks as shallow Gaussian procesesses*, arXiv, 2019. 1808.05587 [stat.ML]. Doi: 10.48550/arXiv.1808.05587

[8] I. Goodfellow, Y. Bengio, A. Courville (2016). *Deep Learning*. The MIT Press, Cambridge MA, U.S.A. http://www.deeplearningbook.org

[9] T. Hazan, T. Jaakkola, *Steps toward deep kernel methods from infinite neural networks*, arXiv, 2015. 1508.05133 [cs.LG]. Doi: 10.48550/arXiv.1508.05133

[10] S. Infante, J. Ortega, F. Cedeño, *Estimación de datos faltantes en estaciones meteorológicas de Venezuela vía un modelo de redes neuronales*, Revista de Climatología **8**, 51–70. Available from: Link

[11] M.E. Khan, A. Immer, E. Abedi, M. Korzepa, *Approximate inference turns deep networks into Gaussian processes*, arXiv, 2019. 1906.01930 [stat.ML]. Doi: 10.48550/arXiv.1906.01930

[12] A. Krizhevsky, I. Sutskever, G. Hinton, *Image net classification with deep convolutional neural networks*, in: F. Pereira, C.J. Burges, L. Bottou & K.Q. Weinberger (Eds.) Advances in Neural Information Processing Systems **25** (2012), 1097–1105. Available from: Link

[13] Y. LeCun, Y. Bengio, G. Hinton, *Deep learning*, Nature **521** (2015), no. 7553, 436–444. Doi: 10.1038/nature14539

[14] J. Lee, Y. Bahri, R. Novak, S.S. Schoenholz, J. Pennington, J. Sohl-Dickstein, *Deep neural networks as Gaussian processes*, arXiv, 2018. 1711.00165v3 [stat.ML]. Doi: 10.48550/arXiv.1711.00165

[15] M. Luca, G. Barlacchi, B. Lepri, L. Pappalardo, *Deep learning for human mobility: a survey on data and models*, arXiv, 2020. 2012.02825 [cs.LG]. Doi: 10.48550/arXiv.2012.02825

[16] D.J.C. MacKay, *Probable networks and plausible predictions. A review of practical Bayesian methods for supervised neural networks*, Network: Computation in Neural Systems, **6** (1992), no. 3, 469–505. Doi: 10.1088/0954-898X_6_3_011

[17] P. Martínez-Arias, R. Fonseca-Delgado, R. Salum, I. Amaro-Martín, *Alzheimer's disease diagnosis system using electroencephalograms and machine learning models*, Iberian Journal of Information Systems and Technologies. (2020) pp. 275–288. Available from: Link

[18] A.G. Matthews, M. Rowland, J. Hron, R.E. Turner, Z. Ghahramani, *Gaussian process behaviour in wide deep neural networks*, arXiv, 2018. 1804.11271v2 [stat.ML]. Doi: 10.48550/arXiv.1804.11271

[19] D. Ming, D. Williamson, S. Guillas, *Deep Gaussian process emulation using stochastic imputation*, arXiv, 2021. 2107.01590 [stat.ML] Doi: 10.48550/arXiv.2107.01590

[20] A. Mosavi, S. Ardabili, A.R. Varkonyi-Koczy, (2020) *List of deep learning models*, in: A.R. Várkonyi-Kóczy (Ed.) Engineering for Sustainable Future, Springer Nature, Cham Switzerland, 2020, pp. 202–214. Doi: 10.1007/978-3-030-36841-8_20

[21] R. Neal, *Bayesian Learning for Neural Networks*, Lecture Notes in Statistics, Springer Verlag, New York, 1996. Doi: 10.1007/978-1-4612-0745-0

[22] R. Novak, L. Xiao, J. Lee, Y. Bahri, G. Yang, J. Hron, ..., J. Sohl-Dickstein, *Bayesian deep convolutional networks with many channels are Gaussian processes*, arXiv, 2019. 1810.05148v4 [stat.ML]. Doi: 10.48550/arXiv.1810.05148

[23] C.E. Rasmussen, C.K.I. Williams, *Gaussian Processes for Machine Learning*, the MIT Press, Cambridge MA, U.S.A., 2006. Available from: http://gaussianprocess.org/gpml/

[24] H. Salimbeni, M. Deisenroth, *Doubly stochastic variational inference for deep Gaussian processes*, arXiv, 2017. 1705.08933v2 [stat.ML]. Doi: 10.48550/arXiv.1705.08933

[25] A. Sauer, R.B. Gramacy, D. Higdon, *Active learning for deep Gaussian process surrogates*, arXiv, 2021. 2012.08015 [stat.ME]. Accepted in Technometrics. Doi: 10.48550/arXiv.2012.08015

[26] S. Schoenholz, J. Gilmer, S. Ganguli, J. Sohl-Dickstein, *Deep information propagation*, arXiv, 2017. 1611.01232v2 [stat.ML]. Doi: 10.48550/arXiv.1611.01232

[27] C.K.I. Williams, *Computing with infinite networks*, in: M.C. Mozer and M. Jordan & T. Petsche (Eds.) Advances in Neural Information Processing Systems **9** (1997), 295–301. Available from: Link

[28] A.G. Wilson, P. Izmailov, *Bayesian deep learning and a probabilistic perspective of generalization*, arXiv, 2020. 2002.08791v3 [cs.LG]. Doi: 10.48550/arXiv.2002.08791

[29] I.H. Witten, E. Frank, M.A. Hall, C.J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 4th Edition. (2016)

[30] Z. Zhao, M. Emzir, S. Särkkä, *Deep state-space Gaussian processes* Statistics and Computing. (2021), 31–75. Doi: 10.1007/s11222-021-10050-6