Deep Gaussian Convolutional Neural Network Model in Classification of Cassava Diseases using Spectral Data

Emmanuel Ahishakiye (ahishema@gmail.com)
Kyambogo University
Waweru Mwangi
Jomo Kenyatta University of Agriculture and Technology
Petronilla Muthoni
Jomo Kenyatta University of Agriculture and Technology
Kanobe Fredrick
Kyambogo University
Godliver Owomugisha
Busitema University
Taremwa Danison
Kyambogo University
Leonard Nkalubo
Kyambogo University

Research Article

Keywords: Gaussian Processes, Crop disease classification, Convolutional Neural Networks, Covariance Functions, Cassava Diseases

Posted Date: June 16th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1750871/v1

Deep Gaussian Convolutional Neural Network Model in Classification of Cassava Diseases using Spectral Data

Emmanuel Ahishakiye*[a, b], Waweru Mwangi[b], Petronilla Muthoni[b], Kanobe Fredrick[c], Godliver Owomugisha[d], Taremwa Danison[c], Leonard Nkalubo[a]

[a]*Department of Networks, Data Science and Artificial Intelligence, Kyambogo University, Kampala, Uganda;*
[b]*Department of Computing, Jomo Kenyatta University of Agriculture and Technology, Kenya;*
[c]*Department of Computer Science, Kyambogo University, Kampala, Uganda*
[c]*Faculty of Engineering, Busitema University, Tororo, Uganda*

Corresponding Author:** *Department of Networks, Data Science and Artificial Intelligence, Kyambogo University, PO BOX 1, Kampala, Uganda;* ***Email*: ahishema@gmail.com;* ***Tel****: +256787371879*

Abstract

Early disease identification in crops is critical for food security, especially in Sub-Saharan Africa. To identify cassava diseases, professionals visually score the plants by looking for disease indicators on the leaves which is notoriously subjective. Automating the detection and classification of crop diseases could help professionals diagnose diseases more accurately and allow farmers in remote locations to monitor their crops without the help of specialists. Machine learning algorithms have been used in the early detection and classification of crop diseases. However, traditional machine learning algorithms are not calibrated even though they have high accuracy. The ability to provide well-calibrated posterior distributions is one of the most appealing properties of Gaussian processes. Motivated by the current developments in the field of Gaussian Processes, this study proposed a deep Gaussian convolutional neural network model (DGCNN) for the detection and classification of cassava diseases using spectral data. The proposed model uses a hybrid kernel function that is the product of a rational quadratic kernel and a squared exponential kernel. Experimental results revealed that our proposed hybrid kernel function performed better in terms of accuracy of 90.10% when compared to both the squared exponential kernel with an accuracy of 88.01% and the rational quadratic kernel with an accuracy of 88.52%. In our future work, we propose to integrate the Optimised model proposed in this study with the transfer learning approach, a move that may help to improve the model performance.

Keywords: *Gaussian Processes, Crop disease classification, Convolutional Neural Networks, Covariance Functions, Cassava Diseases*

1. 1 Introduction

In Sub-Saharan Africa, cassava is the second most significant food crop after maize (1). Because of its persistence in severe locations, as well as its tolerance for extreme environmental stress conditions and poor soils, cassava continues to acquire importance in Africa as a staple food eaten by more than 500 million people every day. However, various pests and diseases, including Cassava Bacterial Blight (CBB), Cassava Brown Steak Disease (CBSD), Cassava Green Mite (CGM), and Cassava Mosaic Disease (CMD), pose a significant danger to crop productivity (1). Early disease identification in crops is critical for food security, especially in Sub-Saharan Africa (2). To identify cassava diseases, government professionals visit various sections of the country and visually score the plants by looking for disease indicators on the leaves (3) which is notoriously

subjective and it is not uncommon for specialists to differ on a plant's diagnosis. Automating the detection and classification of crop diseases could help professionals diagnose diseases more accurately and allow farmers in remote locations to monitor their crops without the help of specialists (3).

Machine learning algorithms have been used in the early detection and classification of crop diseases (4) (5) (6) (7). However, traditional machine learning algorithms are not calibrated even though they have high accuracy (8). In general, decision-makers are concerned not only with predictions but also with the level of confidence in such predictions (9). The ability to provide well-calibrated posterior distributions is one of the most appealing properties of Gaussian processes (10).

The Gaussian process (GP) is a strong but disciplined nonparametric approach that has found wide use in statistical analysis and machine learning. The Gaussian probability distribution is generalized to form a Gaussian process (11). A GP is a type of stochastic process whose property is inherited from the normal distribution, and every finite collection of random variables obeys multivariate normal distribution in probability theory and statistics (12). A stochastic process determines the properties of functions, whereas a probability distribution governs the attributes of scalars or vectors (for multivariate distributions) (13) (14). This probabilistic modeling approach generates not only point estimates, but complete prediction distributions as well. This is especially interesting in light of one of the key goals of agricultural yield estimation: obtaining a credible forecast yield distribution (15).

Gaussian processes (GPs) provide uncertainty estimates as well as a marginal likelihood objective, but their inductive biases result in lower accuracy. Gaussian process models approach supervised learning by assuming a probability distribution over a space of potential functions, updating the space of functions using observed data and applying the Bayes theorem, and calculating the expected value over the space of functions to get an estimate for the function $f(x)$ (16). While Gaussian process models are often utilized in the time series and regression domains, they may also be employed to do classification tasks with the use of a response function and variational inference (17) (18). More so, for issues with short data sets, Gaussian processes (GPs) can be a useful technique. Unlike many supervised machine learning algorithms, such as least squares and Artificial Neural Networks (ANNs), Gaussian Process models are resistant to overfitting and can quantify the uncertainty of predictions (19). Although Gaussian process classifiers perform similarly to non-linear support vector machines (SVMs), they are preferred because of additional benefits such as uncertainty representation and hyper-parameter selection (20). A Deep Gaussian Process (DGP) is a non-parametric deep Bayesian technique based on a hierarchical composition of Gaussian Processes that can overcome the constraints of traditional (single-layer) GPs while keeping their benefits (21) (22). In a deep convolutional Gaussian process, Several GP functions are iteratively convolved over the image (23).

In this study, we used spectral data in a three-class classification challenge for diagnosing cassava diseases. The same data was used in the study (24) with matrix relevance learning techniques. Previous research has used plant image data captured with a smartphone (25) (26) (27) (28). However, disease symptoms must be observable in order to use this strategy (24). Unfortunately, once symptoms appear on the aerial part of the plant, the root, which is the edible part of the plant, is destroyed. By nature, spectral data is exceptionally high-dimensional, with more than 3600

characteristics or dimensions (24). As a dimension reduction strategy, we used conventional principal component analysis (PCA) (29). Motivated by the current developments and many influential studies in the field of Gaussian Processes (30) (31) (23) (18) (32) (21), this study proposes a deep Gaussian convolutional neural network model (DGCNN) for the detection and classification of cassava diseases. By developing a deep Gaussian convolutional neural network model that may improve the crop disease detection accuracy as well as provide uncertainty estimates that are critical for decision-making by farmers, this study is formulated within the research area of Artificial Intelligence for Development (AI4D) and is aimed at contributing to United Nations Sustainable Development Goals (SDGs) number 2, *"End hunger, achieve food security and improved nutrition and promote sustainable agriculture"*.

1.2 The Motivation for use of Gaussian Processes in Classification Tasks

The accuracy of learning algorithms' predicted probabilities can be assessed by looking at how they're calibrated. The calibration of Gaussian Processes has been studied in the literature (22). When a classifier's output accurately represents the likelihood of a given class, that is, when it predicts a given class label with probability p that matches the true proportion p of test points belonging to that class, the classifier is well calibrated (33). The ability to provide well-calibrated posterior distributions is one of the most appealing properties of Gaussian processes (10). Gaussian processes are non-parametric Bayesian models that are effective for prediction modeling. The representation and propagation of uncertainty is an important feature of Bayesian models that is often missed by traditional methodologies. In general, decision-makers are concerned not only with predictions but also with the level of confidence in such projections. Only when the model under evaluation is confident in its forecast, can action be done. This is vital for applications (20) like as medical diagnosis, security, self-driving cars, computer simulations (31), robotics (34), Indoor Positioning Systems (35), spatial-temporal modeling (36), link analysis and transfer learning (37), and system dynamics (38). These uncertainties can be obtained in a rational fashion using Bayesian formalism. Bayesian approaches deal with all types of uncertainty in a model, whether it's in parameter inference or predicting outcomes. Although Gaussian process classifiers perform similarly to non-linear support vector machines (SVMs), some practitioners prefer them because of additional benefits such as uncertainty representation and hyperparameter selection (20). However, the computational and storage complexity of GPs is the main impediment to their use with huge current datasets (39) (40). The issue of computational cost can be addressed by using approximations or iterative matrix computations (41) (42).

1.3 Background on Gaussian Processes

1.3.1 The Gaussian Distribution

The Gaussian or normal distribution is the most extensively used distribution in statistics and machine learning. This is due to several factors. First, it has two simple parameters that capture some of the most fundamental aspects of distribution, namely the mean and variance. Second, the central limit theorem states that sums of independent random variables have a Gaussian distribution, making it an excellent choice for representing residual errors or "noise." Third, subject to the requirement of having a given mean and variance, the Gaussian distribution makes the fewest assumptions (has the highest entropy); this makes it a reasonable default choice in many instances. Finally, it has a straightforward mathematical form that is both simple to execute and highly

effective (43). Given a random variable X, the Gaussian or Normal distribution is written in the form (43) (44):

$$P_X(x) = \mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} exp\left\{-\frac{1}{2\sigma^2}(x - \mu)^2\right\} \quad (1)$$

where $\mu = \mathrm{E}[X]$ is the mean, $\sigma^2 = var[X]$ is the variance and $(2\pi\sigma^2)^{\frac{1}{2}}$ is the normalization constant required to ensure the density integrates to 1 (43). In Equation (1), X represents random variables and x is the real the real argument. The Gaussian or Normal distribution of X is usually represented by $P(x) \sim \mathcal{N}(\mu, \sigma^2)$. For regression tasks, the "functions" formed by linking independent Gaussian vector points are insufficiently smooth; instead, we need these independent Gaussian vector points to be connected as a combined Gaussian distribution. Multivariate normal distribution theory can be used to describe the joint Gaussian distribution (16).

1.3.2 The multivariate Gaussian distribution

The most often used joint probability density function for continuous variables is the multivariate Gaussian or multivariate normal. It's usual for a system to be defined by multiple feature variables $(x_1, x_2, \ldots, x_D)$ that are all correlated. A multivariate Gaussian/normal distribution model is required if we want to model all of these variables as a single Gaussian model. The following is a pdf of a multivariate Gaussian/normal distribution in D dimensions (43) (44):

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}}} \frac{1}{|\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right\} \quad (2)$$

where $\mu = \mathrm{E}[X] \in \mathbb{R}^D$ is a D-dimensional mean vector, x represents the variable, $\Sigma = \mathrm{cov}[\mathrm{x}]$ is a $D\ X\ D$ covariance matrix, and $|\Sigma|$ represents the determinant of Σ. The normalization constant $(2\pi)^{\frac{D}{2}}|\Lambda|^{\frac{1}{2}}$ where $\Lambda = \Sigma^{-1}$ simply ensures that the pdf integrates to 1 (43).

1.3.3 Gaussian Processes

The Gaussian process (GP) is a strong but disciplined nonparametric approach that has found wide use in statistical analysis and machine learning (13). A GP is a type of stochastic process whose property is inherited from the normal distribution, and every finite collection of random variables obeys multivariate normal distribution in probability theory and statistics (12). A stochastic process determines the properties of functions, whereas a probability distribution governs the attributes of scalars or vectors (for multivariate distributions) (13) (14). The Gaussian probability distribution is generalized to form a Gaussian process (11). This probabilistic modeling approach generates not only point estimates, but complete prediction distributions as well (15). Gaussian Process (GP) models approach supervised learning by assuming a probability distribution over a space of possible functions, using observed data to update the space of functions to consider using Bayes theorem and taking the expected value over the space of functions to get an estimate for $f(x)$ (17). A GP defines a prior over functions, which can be turned into a posterior over functions given some data. The prior of GP is denoted by:

$$f(x) \sim GP\big(m(x), k(x, x')\big) \quad (3)$$

where $m(x) = \mathbb{E}[f(x)]$ is the mean function and $k(x, x') = \mathbb{E}\big[\big(f(x) - m(x)\big)\big(f(x') - m(x')\big)\big]$ is the kernel or covariance function and is positive finite (13). A kernel function is defined as a real-valued function with two arguments, $k(x, x') \in \mathbb{R}$, for $x, x' \epsilon \chi$. The function is usually symmetric (i.e., $k(x, x') = k(x', x)$) and non-negative (i.e., $k(x, x') \geq 0$), allowing it to be

interpreted as a measure of similarity (43). More so, function $k(x, x')$ is a kernel function if it is defined purely in terms of inner products in the input space (16). The modeling capability of GPs is determined by the kernel that is selected. In practice, standard stationary kernels result in models that underperform. Because adaptable kernels that account for non-stationary patterns and long-range interactions in the data are difficult to build and infer, shallow or single-layer Gaussian processes are frequently suboptimal (23). For details, refer to (23) (45) (43) (13) (46).

1.3.4 Single-layer Gaussian Processes

Consider the challenge of estimating a stochastic function $f: \mathbb{R}^D \longrightarrow \mathbb{R}$, given also a likelihood $p(y|f)$, and a set of N observations $y = (y_1, \dots, y_N)^T$ at design locations $X = (x_1, \dots, x_N)^T$. A GP prior is placed on the function f that models all function values as jointly Gaussian, with kernel or covariance function $k: \mathbb{R}^D \times \mathbb{R}^D \longrightarrow \mathbb{R}$, and mean function $m: \mathbb{R}^D \longrightarrow \mathbb{R}$. Also, a set of M inducing locations is defined as $Z = (z_1, \dots, z_M)^T$. The function values at the design and inducing points are, $f = f(X)$ and $u = f(Z)$ respectively. Also, let $[m(X)]_i = m(x_i)$ and $[k(X, Z)]_{ij} = k(x_i, z_j)$. Considering the definition of GP, the joint density $p(f, u)$ is a Gaussian with a mean determined by the mean function at each input $(X, Z)^T$, and also the covariance function is evaluated at each pair of inputs to yield the corresponding covariance. The joint density of $y, f \; and \; u$ is defined as (22):

$$p(y, f, u) = \big(p(f|u; X, Z)p(u; Z)\big)\Big(\prod_{i=1}^{N} p(y_i|f_i)\Big) \qquad (4)$$

The first term is the GP prior and the second term is the likelihood. The GP prior $p(f, u; X, Z)$ is factorized into prior $p(u) = \mathcal{N}\big(\mathrm{u}\big|\mathrm{m(Z)}, \mathrm{k(Z, Z)}\big)$ and the conditional $p(f|u; X, Z) = \mathcal{N}(f|\mu, \Sigma)$, and for $i, j = 1, \dots, N$.

$$[\mu]_i = m(x_i) + \alpha(x_i)^T\big(u - m(Z)\big), \qquad (5)$$

$$[\Sigma]_{\mathrm{ij}} = k\big(x_i, x_j\big) - \alpha(x_i)^T k(\mathrm{Z, Z})\alpha\big(x_j\big), \qquad (6)$$

where $(x_i) = k(\mathrm{Z, Z})^{-1}k(Z, x_i)$.

1.3.5 Deep Gaussian Processes

A Deep Gaussian Process (DGPs) is a hierarchical composition of Gaussian Processes that can overcome the constraints of traditional (single-layer) GPs while keeping their benefits. Deep networks are richer than generalized linear models, and Deep Gaussian Processes (DGPs) are richer than regular GPs. DGPs train a representation hierarchy non-parametrically with extremely few hyper-parameters to tune, in contrast to models with extensively parameterized kernels (22). A Deep Gaussian Process model recursively defines a prior on vector-valued stochastic functions $F^1, \dots, F^L$ (47). The prior on each function F^l is an independent GP in each dimension, with input locations determined by the noisy corruptions of the function values at the following layer: the outputs of the GPs at layer l are F_d^l, and the corresponding inputs are F^{l-1}. The study (22) approached this issue by not parameterizing these variables separately but instead absorbed the noise into the kernel $k_{noisy}\big(x_i, x_j\big) = k\big(x_i, x_j\big) + \sigma_l^2\delta_{ij}$, where δ_{ij} is the kronecker delta and σ_l^2 is noise variance between layers. Let D^l be the dimension of the outputs at layer l. The inducing locations is Z^{l-1} at each layer and the inducing function values is U^l for each dimension which is the case as with single layer case. The joint density is defined as follows (22):

$$p(Y, \{F^l, U^l\}_{l=1}^L) = \Big(\prod_{i=1}^{N} p(Y_i|\, f_i^L)\Big)\prod_{l=1}^{L} p(F^l|U^l;\, F^{l-1}, Z^{l-1})p(U^l;\, Z^{l-1}), \quad (7)$$

where the first term is the likelihood and the second term is the DGP prior. Also, $F^0 = X$. In this model, the inference is difficult, hence approximations must be utilized (Salimbeni & Deisenroth, 2017; Bui et al., 2016; Damianou & Lawrence, 2013).

1.3.6 Integrating Neural Networks with Deep Gaussian Processes

Deep Gaussian processes are more versatile and powerful than shallow GPs but they produce degenerate models if the individual GP layers are not invertible, limiting their potential (23). Deep Gaussian processes improve performance by simulating GP node networks (49). Deep Gaussian processes (DGPs) are multi-layer generalizations of Gaussian processes, however, inference has proven difficult in these models (22). Deep Gaussian processes, though more versatile and powerful than shallow GPs, produce degenerate models if the individual GP layers are not invertible, limiting their potential (23). In a deep convolutional Gaussian process, Several GP functions are iteratively convolved over the image (23). The study (39) revealed that Deep Gaussian Processes (DGPs) can handle a lot more layers than is commonly assumed in the literature on DGPs, and they're also scalable to big datasets where DGP inference was previously thought to be impossible. To improve their performance, GPs models are being combined with neural networks to come up with robust models. The study (12) revealed that Deep neural networks' interpretability can be studied statistically by replacing the special layer with a Gaussian process model (12). More so, the study (50) extended the convolutional network to the state of knowledge regarding GPs and Deep networks. A method for incorporating a Deep GP as the output layer of an ANN has been recently proposed (51). It is based on the following two premises (52):

i. The ANN computes a mapping net of n_I-dimensional input values into the set X on which the GP is defined if $n_1$specifies the number of ANN input neurons. As a result, the ANN transforms an input v into a point $x = net(v) \in X$, corresponding to an observation $f(x) + \epsilon$ controlled by the GP, and the number n_O of neurons in the last hidden layer fulfills $X \subset \mathbb{R}^{n_O}$. The GP is now $GP(m_{GP}(net(v)), k(net(v), net(\acute{v}))$ from the perspective of the ANN inputs.
ii. The GP mean m_{GP} is considered to be a known constant, thus having no effect on the GP hyperparameters and being independent of the net.

The GP is solely determined by the covariance function's parameters θ^k. The ANN is determined by the vector θ^W of its weights and biases on the one hand, and the network architecture on the other, which are considered fixed before network training. Consider the n inputs to the neural network, $v_1, \dots, v_n$, which correspond to the GP inputs $x_1 = net(v_1), \dots, x_n = net(v_n)$, which correspond to the observations $y = (y_1, \dots, y_n)^T$. The loglikelihood of $\theta = (\theta^k, \theta^W)$ is then;

$$\mathcal{L}(\theta) = \ln p(y; m_{GP}, k, \sigma_n^2)$$

$$\mathcal{L}(\theta) = -\frac{1}{2}(y - m_{GP})^T K^{-1}(y - m_{GP}) - \ln(2\pi) - \frac{1}{2}\ln det\,(K + \sigma_n^2 I_n) \qquad (8)$$

Where m_{GP} is mean and constant and

$$(K)_{i,j} = k\left(net(v_i), net(v_j)\right) \qquad (9)$$

The procedure of searching for the vector (θ^k, θ^W) during model training is done using gradient descent. The partial derivatives that makeup $\nabla_{(\theta^k, \theta^W)}\mathcal{L}$ can be calculated as follows:

$$\frac{\partial \mathcal{L}}{\partial \theta_\ell^k} = \sum_{i,j=1}^{n} \frac{\partial \mathcal{L}}{\partial K_{i,j}} \frac{\partial K_{i,j}}{\partial \theta_\ell^k}, \qquad (10)$$

$$\frac{\partial \mathcal{L}}{\partial \theta_\ell^W} = \sum_{i,j,k=1}^{n} \frac{\partial \mathcal{L}}{\partial K_{i,j}} \frac{\partial K_{i,j}}{\partial x_k} \frac{\partial net(v_k)}{\partial \theta_\ell^W}. \quad (11)$$

The partial derivatives $\frac{\partial \mathcal{L}}{\partial K_{i,j}}, i, j = 1, \ldots, n,$ are the components of the derivative $\frac{\partial \mathcal{L}}{\partial K}$ which yields;

$$\frac{\partial \mathcal{L}}{\partial K} = \frac{1}{2}(K^{-1} y y^T K^{-1} - K^{-1}) \quad (12)$$

2.1 A Premier on some of the Covariance or Kernels Functions

(i) The squared exponential Kernel

The squared exponential (SE) kernel function (53) (54) (43) (13) is the most extensively used kernel or covariance function. For Gaussian processes, the SE kernel is the de-facto default kernel. This is owing to its universal nature, which allows it to be integrated with most functions. The SE kernel is also called the radial basis function (RBF) kernel or the Gaussian kernel function. The SE Kernel is defined in Equation 13 as:

$$k_{SE}(x, x') = \sigma^2 exp\left(-\frac{(x - x')^2}{2\ell^2}\right) \quad (13)$$

where the two parameters are: The 'wiggles' in the function are determined by the lengthscale ℓ. It is often difficult to extrapolate more than ℓ units from the data in most cases. The average distance of the function from its mean is determined by the output variance σ^2. This is a scaling factor that appears in every kernel.

(ii) The Matern 5/2 kernel

The SE kernel generates infinitely differentiable, smooth functions. For many applications, the Matern kernel (53) (54) (43) (13) is preferable because it generates "rougher" functions that can better mimic local "wiggles" without requiring a very small overall length scale. The Matern kernel is often used in Gaussian process regression. The Matern 5/2 Kernel is defined in Equation 14 as:

$$k_{M52}(x, x') = \sigma\left(1 + \sqrt{5r} + \frac{5}{3}r^2\right)\exp\left(-\sqrt{5r}\right) \; where \; r = \frac{\|x - x'\|_2}{\ell} \quad (14)$$

where ℓ is the lengthscale and σ is the standard deviation.

(iii) The Rational Quadratic Kernel

The Rational Quadratic Kernel (RQ) (53) (54) (43) (13) is the result of combining many SE kernels with varying lengthscales. As a result, GP priors with this kernel expect to see smooth functions over a wide range of lengthscales. The parameter α controls how large-scale and small-scale fluctuations are weighted. When α→∞, the RQ and the SE are the same. The RQ kernel is defined in Equation 15 as:

$$K_{RQ}(x, x') = \sigma^2\left(1 + \frac{(x - x')^2}{2\alpha\ell^2}\right)^{-\alpha} \quad where \quad \alpha > 0 \quad (15)$$

(iv) The Periodic Kernel

The periodic kernel (53) (54) (43) (13) can be used to model functions that exactly repeat themselves. Its parameters are simple to understand: The periodic kernel is defined in Equation 16 as:

$$k_{Per}(x, x') = \sigma^2 exp\left(-\frac{2\sin^2(\pi|x - x'|/p}{\ell^2}\right) \quad (16)$$

The distance between repetitions of the function is determined by the period p. In the same way as the SE kernel, the lengthscale ℓ determines the lengthscale function.

(iv) The Locally Periodic Kernel

The majority of periodic functions do not exactly repeat themselves. To give our model more flexibility, we can add or multiply a local kernel, such as the squared exponential, with our periodic kernel (53) (54) (43) (13). This allows us to describe functions that are only locally periodic; the shape of the function's repeating component can now change over time. The locally periodic kernel is defined in Equation 17 as:

$$k_{LPer}(x, x') = \sigma^2 exp\left(-\frac{2\sin^2(\pi|x - x'|/p}{\ell^2}\right) exp\left(-\frac{(x - x')^2}{2\ell^2}\right) \quad (17)$$

2.2 Designing New Covariance or Kernels Functions

The conventional kernels above are excellent if all of your data is of the same type, but what if you have multiple types of features and want to regress on all of them at the same time? (54) A new kernel may be designed that suits that at-hand data well. Several studies (53) (54) (43) (13) proposed ways of building new kernels from existing kernels. The conventional technique to create new kernels is to multiply them together, especially if they are defined on different inputs to the function. Multiplying two kernels is roughly equivalent to an AND operation. That is, if you multiply two kernels together, the final kernel will only have a high value if both of the base kernels are high. Given two valid kernels $K_1(x, \acute{x})$ and $K_2(x, \acute{x})$, the product of two kernels is a kernel (Equation 18) (53) (54) (43) (13).

$$k_1 \times k_2 = k_1(x, \acute{x}) \times k_2(x, \acute{x}) \quad (18)$$

Proof:

Let $f(x) = f_1(x)f_2(x)$ be a random process, where $f_1(x)\ and\ f_2(x)$ are independent. Then, $k(x, \acute{x}) = k_1(x, \acute{x})\ k_2(x, \acute{x})$ is a covariance function. An extension of this argument reveals that $k^q(x, \acute{x})$ is a valid covariance function for $q \in \mathbb{N}$ (13). For further details on kernel designs, refer to (53) (54) (43) (13).

3.1 Materials and Methods

3.1.1 Image Dataset

Spectral data (24) was obtained from the leaves of cassava plants in two conditions: when they were healthy and when they were infected with the two diseases CBSD and CMD, both of which had visible symptoms. A CI-710 miniature leaf spectrometer was used to collect this data (CID BioScience Inc 2010). The device is powered by USB from a device (such as a tablet or laptop), allowing the setup to be mobile and collect data in the field. The device is clamped onto a leaf of a certain plant to gather data, and the profile of the amount of light absorbed or reflected is captured

on the device as a spectrogram for each position of the clamped leaf. The intensity and structure of the spectra are influenced by a variety of environmental conditions, with illumination being particularly relevant. As a result, data was gathered in the field under identical lighting conditions. Data was collected for plants aged 6 to 9 months from Nase 3, Nase 4, Nase 14, Nase 19, Alado Alado, Magana, Orera, and NAROCass 2 cassava types. Three plants were selected for each variety, and three leaves were chosen for each plant. On each leaf lobe, two spectral measurements were taken: one on the best (least affected/non-symptomatic) section and one on the worst (most affected/symptomatic) section (see Figure 1). Because the spectrometer only measures a small area of the plant, around 2cm in diameter, readings for each leaf lobe were taken to ensure a representative and reliable sample. The dataset was collected in Uganda with the guidance of agricultural and bio-chemical experts from the National Crops Resources Research Institute (NaCRRI)[1]. The National Agricultural Research and Development Research Institute (NaCRRI) is the Ugandan government's entity in charge of agricultural research. A total of 1656 data points were collected across three classes: healthy, CMD, and CBSD. For details, refer to (24). Figure 2 shows an example of a dataset for each class of cassava leaves.

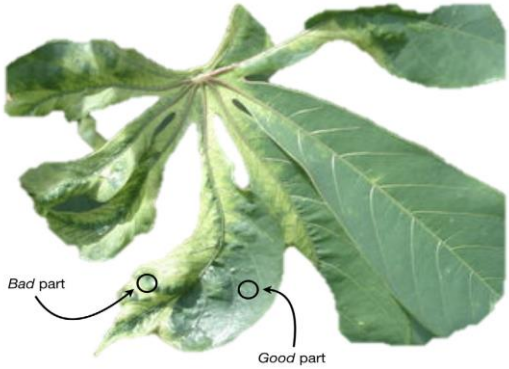

Figure 1: Depiction of asymptomatic(good) and symptomatic(bad) parts of the plant. Adapted from (24).

Figure 2: Cassava image data with 3 categories used in this study, Adopted from (24).

3.1.2 The Proposed Covariance Function/Kernel

In this section, we discuss the proposed covariance function that was used with the DGP model. The proposed covariance function (Equation 22) is the product of a rational quadratic kernel (Equation 20) and a squared exponential kernel also called the radial basis function (RBF) kernel or the Gaussian kernel function (Equation 19). The parameters for the covariance function used in each of the experiments are; length-scale (ℓ), which encodes the "wiggliness" of the GP, and

[1] www.naro.go.ug

variance (σ^2), which tunes the amplitude (43) (13) (44). The proposed covariance function was implemented using GPflux (55) and GPflow (56). We used default parameters in all our experiments.

We discuss how we developed our proposed covariance function in Equations 18 - 22. Given two valid kernels $K_1(x, \acute{x})$ and $K_2(x, \acute{x})$, the product of two kernels is a kernel (Equation 18) (53) (54) (13). In this study, we propose a hybrid covariance function that is a product of a rational quadratic kernel and a squared exponential kernel using the concept in Equation 18. The mathematical expressions for a rational quadratic kernel and a squared exponential kernel are shown in Equations 14 and 15 respectively.

$$k_{SE}(x, x') = \sigma^2 exp\left(-\frac{(x - x')^2}{2\ell^2}\right) \tag{19}$$

$$K_{RQ}(x, x') = \sigma^2 \left(1 + \frac{(x - x')^2}{2\alpha\ell^2}\right)^{-\alpha} \quad where \quad \alpha > 0 \tag{20}$$

Therefore, from Equation 18;

$$K_{Hybrid}(x, x') = K_{RQ}(x, x') \times k_{SE}(x, x') \tag{21}$$

$$K_{Hybrid}(x, x') = \frac{\sigma^2}{2\ell^2}\left[\left(\frac{2\alpha\ell^2 + (x - x')^2}{\alpha}\right)^{-\alpha} \exp(-(x - x')^2)\right] \tag{22}$$

where ℓ is the length scale, σ^2 is the output variance, and the weighting of large-scale and small-scale fluctuations is determined by the parameters α. For details on kernel design, refer to (43) (13) (44).

3.1.3 The Proposed Model

In this study, we propose a hybrid model (DGCNN) that combines a deep Gaussian process (DGP) model and a convolutional neural network (CNN). We started by building a DGP model using the python libraries GPflow (56) and GPflux (55). The DGP model was comprised of the following; kernel function, the inducing variable, the GP-layer, and the likelihood layer (The likelihood layer is in charge of computing the objective's variational expectation as well as dealing with our likelihood distribution $p(y/f)$. During this study, we used Gaussian likelihood) and the likelihood container. The DGP model was implemented using DeepGP in the GPflux library. The DeepGP is the specialization of the Keras model. Keras is in charge of data minibatching and keeping track of our training losses. After the DGP model, we then implemented the final model (DGCNN), which is a CNN model with the DGP model as the prediction layer. The DGCNN is comprised of 6 dense layers and the final prediction layer is the DGP model. Also, spectral data is exceptionally high-dimensional, with more than 3600 characteristics or dimensions (24). We used a standard scaler (57) for scaling the dataset and conventional principal component analysis (PCA) (29) was used as a dimension reduction strategy. The pseudocode of the proposed model is given in Figure 2 below.

Pseudocode for DGCNN Model

```
Input: Model parameters
Output: Prediction prediction
Start:
        Load data
        Standardize data using a standard scaler
        Dimensionality reduction using PCA
Build Deep GP Model
        Build the kernel
        Define the inducing variable
        Define the likelihood layer
        Define the likelihood container
        Build the Deep GP_layer
Integrate Deep GP Model with CNN
        Define a sequential CNN model with the following layers in order
                6 Dense layers
                1 linear layer
                Deep GP_layer as the prediction layer
Model compilation
Model fitting
Prediction results
```

Figure 3: The Pseudocode for the proposed DGCNN Model

3.1.4 Model Evaluation

The model was evaluated using accuracy and loss function. The accuracy is given by Equation 23.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (23)$$

Where TP is true positive, TN is true negative, FP is false positive and FN is false negative.

The loss function, calling it yields the negative variational expectation $-\mathrm{E}_{q(f)}[\log p(y/f)]$ when the prediction (last-layer output) is a Distribution q(f). This loss returns the negative log-likelihood $-\log p(y/f)$ when the prediction is a tf.Tensor. The value of this loss is not explicitly logged when you use this loss function to train a Keras model (in contrast, the layer-specific losses are logged, as is the overall model loss). In this study, we used this loss function during the training of our proposed model which was implemented in Keras.

3.1.5 Model Implementation

During this study, experiments were done in google colab[2]. Also, a windows 10 laptop, Intel Core i7 processor of 2.50 GHz, and 16GB RAM with 2GB of NVIDIA GeForce MX150 graphical processing unit (GPU) was used. Also, the python libraries GPflow (56) and GPflux (55) were used.

[2] https://colab.research.google.com/?utm_source=scs-index

4.1 Experimental Results and Discussion

In this section, we present the results of the experiments. We initially set ℓ and σ to 1. We also used the batch size of 5 since gaussian processes suffer from a cubic time complexity $O(n^3)$ since the inversion and determinant of the $n \times n$ kernel matrix $K_{nn} = k(X, X)$ which limits the scalability and unaffordable of GPs on large datasets (58). Since the model integrates deep Gaussian processes with convolutional neural networks, the optimizer used was Adam (59) and we used its default learning rate. Experiments were done using squared exponential kernel (Table 1), rational quadratic kernel (Table2), and our proposed hybrid kernel (Table 3). Experimental results revealed that our proposed hybrid kernel function performed better in terms of accuracy and loss when compared to both the squared exponential kernel and rational quadratic kernel as shown in Table 3. Results also revealed that the rational quadratic kernel (Table 2) performed better when compared to squared exponential kernel (Table 1). The results were recorded with several epochs and in all, our proposed hybrid kernel function performed better compared to the other kernel functions. For example, after 500 epochs, the proposed hybrid kernel had accuracy of 90.10%, followed by rational quadratic kernel with accuracy of 88.52% and then squared exponential kernel with accuracy of 88.01%. In terms of validation loss after 500 epochs, the proposed hybrid kernel had 0.0105, followed by rational quadratic kernel with 0.0316 and then squared exponential kernel with 0.0444. This reveals that our proposed kernel function had lower loss value when compared to rational quadratic kernel and squared exponential kernel.

Table 1: Experimental Results using Squared Exponential Kernel

Epoch	Accuracy (%)	Loss
100	85.44	0.2729
200	86.17	0.2101
300	86.93	0.1509
400	87.55	0.0897
500	88.01	0.0444

Table 2: Experimental Results using Rational Quadratic Kernel

Epoch	Accuracy (%)	Loss
100	85.93	0.2450
200	86.48	0.1782
300	87.20	0.1211
400	87.84	0.0647
500	88.52	0.0316

Table 3: Experimental Results using the Hybrid Kernel [Proposed]

Epoch	Accuracy (%)	Loss
100	87.22	0.2106
200	87.89	0.1282
300	88.56	0.0701
400	89.33	0.0317
500	90.10	0.0105

4.2 Comparative Performance of the Kernel functions on Accuracy and Loss

Experiments were done with our proposed model to determine the performance of the kernel functions and the results were plotted as shown in Figure 3. It can be revealed that the proposed kernel function performed better in terms of accuracy (left) and training loss (right). The results further show that the squared exponential kernel had the least performance when compared to the three kernel functions.

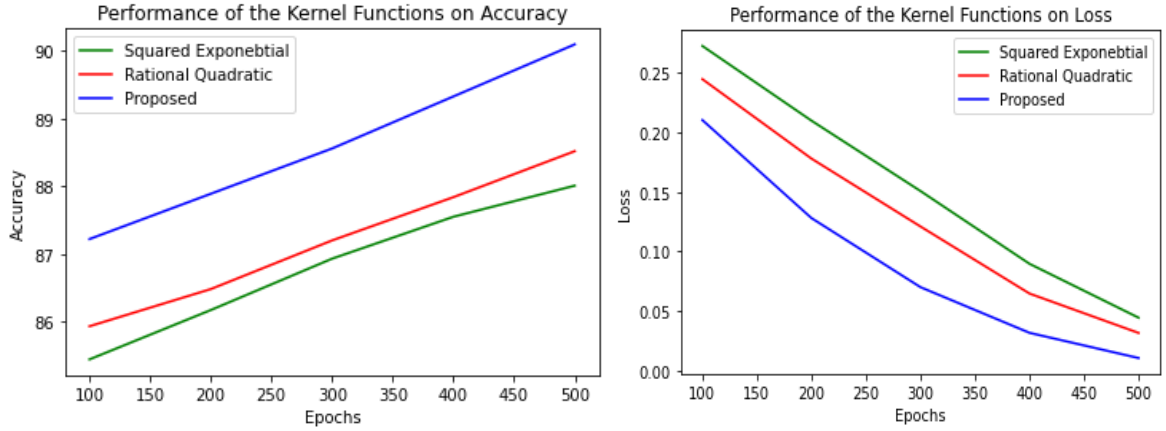

Figure 3: Performance of the kernel functions in terms of Accuracy (Left) and Loss (Right)

5.1 Conclusion

Machine learning algorithms have been used in the early detection and classification of crop diseases. However, traditional machine learning algorithms are not calibrated even though they have high accuracy. The ability to provide well-calibrated posterior distributions is one of the most appealing properties of Gaussian processes. Motivated by the current developments in the field of Gaussian Processes, this study proposed a Deep Gaussian convolutional neural network model (DGCNN) for the detection and classification of cassava diseases that integrates Deep Gaussian Processes with a convolutional neural network. The proposed model uses a hybrid Kernel function that is the product of a rational quadratic kernel and a squared exponential kernel. Experimental results revealed that our proposed hybrid kernel function performed better in terms of accuracy and loss when compared to both the squared exponential kernel and rational quadratic kernel. Experimental results after 500 epochs revealed that our proposed hybrid kernel function performed better in terms of accuracy of 90.10% when compared to both the squared exponential kernel with an accuracy of 88.01% and the rational quadratic kernel with an accuracy of 88.52%. In future work, we propose to integrate the Optimised model proposed in this study with the transfer learning approach, a move that may help to improve the model performance.

Abbreviations

DGCNN: Deep Gaussian convolutional neural network; ML: machine learning; CNN: convolutional neural networks; DL: deep learning; AI: Artificial Intelligence; AI4D: Artificial Intelligence for Development.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials
The data and the code used during this study will be shared on request.

Competing interests
The authors declare that they have no competing interests.

Funding
This research was made possible through DAAD. The authors gratefully acknowledge DAAD for their financial support.

Authors' contributions
Emmanuel Ahishakiye (EA), Waweru Mwangi (WM), Petronilla Muthoni (PM), Kanobe Fredrick (KF), Godliver Owomugisha (GO), Taremwa Danison (TD), and Leonard Nkalubo (LN) designed the study. EA wrote the manuscript. WM, PM, KF, GO, TD, and LK edited and extensively reviewed the manuscript. All the authors approved the manuscript.

Acknowledgments
Not applicable.

References

1. Owomugisha G. Computational intelligence & modeling crop disease data in Africa. University of Groningen; 2020.
2. Aryee SND, Owusu-Adjei D, Osei-Amponsah R, Skinner B, Sowatey E, Sargent CA. Sustainable genomic research for food security in sub-Saharan Africa. Agric Food Secur [Internet]. 2021;10(1):1–12. Available from: https://doi.org/10.1186/s40066-021-00287-9
3. Mwebaze E, Gebru T, Frome A, Nsumba S, Tusubira J. iCassava 2019 Fine-Grained Visual Categorization Challenge. arXiv:190802900v2 [Internet]. 2019; Available from: http://arxiv.org/abs/1908.02900
4. Liu J, Wang X. Plant diseases and pests detection based on deep learning: a review. Plant Methods [Internet]. 2021;17(1):1–18. Available from: https://doi.org/10.1186/s13007-021-00722-9
5. Goyal L, Sharma CM, Singh A, Singh PK. Leaf and spike wheat disease detection & classification using an improved deep convolutional architecture. Informatics Med Unlocked [Internet]. 2021;25(April):100642. Available from: https://doi.org/10.1016/j.imu.2021.100642
6. Rangarajan Aravind K, Raja P. Automated disease classification in (Selected) agricultural crops using transfer learning. Automatika [Internet]. 2020;61(2):260–72. Available from: https://doi.org/10.1080/00051144.2020.1728911
7. Mohanty SP, Hughes DP, Salathé M. Using deep learning for image-based plant disease detection. Front Plant Sci. 2016;7(September):1–10.
8. Lecun Y, Bottou L, Bengio Y, Ha P. Gradient-Based Learning Applied to Document Recognition. Proc IEEE. 1998;(November):1–46.
9. Koriyama T. An introduction of Gaussian processes and deep Gaussian processes and their applications to speech processing. Acoust Sci Technol. 2020;41(2):457–64.
10. Pleiss G, Gardner JR, Weinberger KQ, Wilson AG. Constant-time predictive distributions for Gaussian processes. 35th Int Conf Mach Learn ICML 2018. 2018;9:6575–84.

11. Rasmussen CE. Gaussian Processes in machine learning. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics). 2004;3176:63–71.
12. Zhang B. Gaussian processes based transfer learning for online multiple-person tracking and building blocks for deep learning. Brunel University London; 2020.
13. Rasmussen CE, Williams CKI. Gaussian processes for machine learning. Vol. 14. London, England: The MIT Press; 2006. 243 p.
14. Williams CKI, Barber D. Bayesian classification with gaussian processes. IEEE Trans Pattern Anal Mach Intell. 1998;20(12):1342–51.
15. Wu W, Wu X, Zhang YY, Leatham D. Gaussian process modeling of nonstationary crop yield distributions with applications to crop insurance. Agric Financ Rev. 2021;
16. Wang J. An Intuitive Tutorial to Gaussian Processes Regression [Internet]. Kingston, ON K7L 3N6 Canada; 2021. Available from: http://arxiv.org/abs/2009.10862
17. Frank HT. Gaussian Process Models for Computer Vision. California State Polytechnic University, Pomona; 2020.
18. Görtler J, Kehlbeck R, Deussen O. A Visual Exploration of Gaussian Processes [Internet]. 2019 [cited 2021 Jun 30]. Available from: https://distill.pub/2019/visual-exploration-gaussian-processes/
19. Sit H, Earls CJ. Gaussian Process Regression for Estimating EM Ducting Within the Marine Atmospheric Boundary Layer. Radio Sci [Internet]. 2020;55(6):1–14. Available from: https://doi.org/10.1029/2019RS006890
20. Wistuba M, Rawat A. Scalable Large Margin Gaussian Process Classification. In: ECML PKDD 2019, LNAI 11907. Springer Nature Switzerland AG 2020; 2020. p. 501–516.
21. Kumar V, Singh V, Srijith PK, Damianou A. Deep Gaussian processes with convolutional kernels. arXiv. 2018;
22. Salimbeni H, Deisenroth MP. Doubly stochastic variational inference for deep Gaussian processes. Adv Neural Inf Process Syst. 2017;2017-Decem(Nips):4589–600.
23. Blomqvist K, Kaski S, Heinonen M. Deep Convolutional Gaussian Processes. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics). 2020;11907 LNAI:582–97.
24. Owomugisha G, Melchert F, Mwebaze E, Quinn JA, Biehl M. Matrix Relevance Learning from Spectral Data for Diagnosing Cassava Diseases. IEEE Access. 2021;9:83355–63.
25. Abayomi-Alli OO, Damaševičius R, Misra S, Maskeliūnas R. Cassava disease recognition from low-quality images using enhanced data augmentation model and deep learning. Expert Syst. 2021;38(7):1–21.
26. Lilhore UK, Imoize AL, Lee CC, Simaiya S, Pani SK, Goyal N, et al. Enhanced Convolutional Neural Network Model for Cassava Leaf Disease Identification and Classification. Mathematics. 2022;10(4).
27. Ramcharan A, Baranowski K, McCloskey P, Ahmed B, Legg J, Hughes DP. Deep learning for image-based cassava disease detection. Front Plant Sci. 2017;8(October):1–7.
28. Ramcharan A, McCloskey P, Baranowski K, Mbilinyi N, Mrisho L, Ndalahwa M, et al. A mobile-based deep learning model for cassava disease diagnosis. Front Plant Sci. 2019;10(March):1–8.
29. Reddy GT, Reddy MPK, Lakshmanna K, Kaluri R, Rajput DS, Srivastava G, et al. Analysis of Dimensionality Reduction Techniques on Big Data. IEEE Access. 2020;8:54776–88.
30. Dutordoir V, van der Wilk M, Artemev A, Hensman J. Bayesian Image Classification with Deep Convolutional Gaussian Processes. In: Proceedings of the 23rdInternational

Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy [Internet]. 2019. Available from: http://arxiv.org/abs/1902.05888
31. Sauer A, Gramacy RB, Higdon D. Active Learning for Deep Gaussian Process Surrogates [Internet]. 2020. Available from: http://arxiv.org/abs/2012.08015
32. Dutordoir V, Salimbeni H, Deisenroth MP, Hensman J. Gaussian process conditional density estimation. Adv Neural Inf Process Syst. 2018;2018-Decem(NeurIPS):2385–95.
33. Tran GL, Cunningham JP, Bonilla E V., Michiardi P, Filippone M. Calibrating Deep Convolutional Gaussian Processes. Proc 22nd Int Conf Ar- tificial Intell Stat 2019, Naha, Okinawa, Japan. 2019;89.
34. Wagle N, Frew EW. Forward Adaptive Transfer Using Gaussian Processes. J Aerosp Inf Syst. 2017;14(4).
35. Xie Y, Zhu C, Jiang W, Bi J, Zhu Z. Analyzing Machine Learning Models with Gaussian Process for the Indoor Positioning System. Math Probl Eng. 2020;2020.
36. Reece S, Roberts S. An introduction to Gaussian processes for the Kalman filter expert. 13th Conf Inf Fusion, Fusion 2010. 2010;
37. Yu K, Chu W. Gaussian process models for link analysis and transfer learning. Adv Neural Inf Process Syst 20 - Proc 2007 Conf. 2009;1–8.
38. Bosch N, Achterhold J, Leal-Taixé L, Stückler J. Planning from Images with Deep Latent Gaussian Process Dynamics. arXiv. 2020;
39. Cutajar K. Broadening the Scope of Gaussian Processes for Large-Scale Learning. 2019;(April 2019):13. Available from: https://www.eurecom.fr/publication/5852
40. Snelson E, Ghahramani Z. Sparse Gaussian processes using pseudo-inputs. Adv Neural Inf Process Syst. 2005;1257–64.
41. Neal RM. Regression and Classification Using Gaussian Process Priors. Bayesian Stat. 1998;6:475–501.
42. Gibbs MN. Bayesian Gaussian Processes for Regression and Classification. Thesis [Internet]. 1997;134. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.147.1130&rep=rep1&type=pdf
43. Murphy KP. Machine Learning: A Probabilistic Perspective. London, England: The MIT Press; 2012. 1067 p.
44. Bishop CM. Pattern Recognition and Machine Learning. 1st ed. Springer-Verlag New York; 2006. 738 p.
45. Wilk M Van Der, Rasmussen CE, Hensman J. Convolutional Gaussian Processes. 2017;(Nips):1–10.
46. Seeger M. Gaussian processes for machine learning. Vol. 14, International journal of neural systems. 2004. 69–106 p.
47. Damianou AC, Lawrence ND. Deep Gaussian processes. J Mach Learn Res. 2013;31:207–15.
48. Bui TD, Hernández-Lobato JM, Hernández-Lobato D, Li Y, Turner RE. Deep Gaussian processes for regression using approximate expectation propagation. 33rd Int Conf Mach Learn ICML 2016. 2016;3:2187–208.
49. Dormann CF. Calibration of probability predictions from machine-learning and statistical models. Glob Ecol Biogeogr. 2020;29(4):760–5.
50. Borovykh A. A Gaussian Process perspective on Convolutional Neural Networks [Internet]. 2019. Available from: http://arxiv.org/abs/1810.10798
51. Wilson AG, Hu Z, Salakhutdinov R, Xing EP. Deep kernel learning. Proc 19th Int Conf Artif

Intell Stat AISTATS 2016. 2016;51:370–8.
52. Růžička J, Koza J, Tumpach J, Pitra Z, Holeňa M. Combining Gaussian Processes with Neural Networks for Active Learning in Optimization. CEUR Workshop Proc. 2021;3079:105–20.
53. Murphy KP. Probabilistic Machine Learning: Advanced Topics. Dietterich T, Bishop C, Heckerman D, Jordan M, Kearns M, editors. The MIT Press Cambridge, Massachusetts London, England; 2022. 1294 p.
54. Duvenaud DK. Automatic Model Construction with Gaussian Processes [Internet]. PhD Thesis, University of Cambridge. 2014. Available from: https://www.repository.cam.ac.uk/handle/1810/247281%0Ahttps://www.cs.toronto.edu/~duvenaud/thesis.pdf
55. Dutordoir V, Salimbeni H, Hambro E, McLeod J, Leibfried F, Artemev A, et al. GPflux: A Library for Deep Gaussian Processes. 2021; Available from: http://arxiv.org/abs/2104.05674
56. Matthews DG, Alexander G, Van Der Wilk M, Nickson T, Fujii K, Boukouvalas A, et al. GPflow: A Gaussian process library using TensorFlow. J Mach Learn Res [Internet]. 2017;18(40):1–6. Available from: http://jmlr.org/papers/v18/16-537.html%0Ahttps://www.gpflow.org/%0Ahttps://github.com/GPflow/GPflow
57. Ahsan MM, Mahmud MAP, Saha PK, Gupta KD, Siddique Z. Effect of Data Scaling Methods on Machine Learning Algorithms and Model Performance. Technologies. 2021;9(3):52.
58. Liu H, Ong YS, Shen X, Cai J. When Gaussian Process Meets Big Data: A Review of Scalable GPs. IEEE Trans Neural Networks Learn Syst. 2020;31(11):4405–23.
59. Kingma DP, Ba JL. Adam: A method for stochastic optimization. 3rd Int Conf Learn Represent ICLR 2015 - Conf Track Proc. 2015;1–15.