Curse of Smoothness in Functional Neural Networks

Taehun Cha and Donghun Lee

***Abstract*—Functional neural networks (FNNs) have emerged as powerful tools for modeling complex relationships in functional data, leveraging the flexibility of deep learning to capture non-linear patterns. However, most components of FNNs are directly borrowed from standard deep neural networks, such as element-wise non-linear activation functions and gradient-based optimization strategies. In this study, we investigate how the functional nature of FNNs affects gradient-based optimization. Analogous to the well-known vanishing gradient problem, we theoretically show that the smoothness of the hidden state function bounds the weight gradient norm, a phenomenon we call the *curse of smoothness*. Empirically, we demonstrate that FNN optimization becomes significantly more difficult as model depth increases, compared to conventional deep neural networks. In particular, we verify that gradients in FNNs vanish in deeper layers as the hidden state functions become smoother. These findings suggest that applying standard deep learning techniques to functional data without accounting for the unique properties of functional data can lead to misleading or suboptimal results.**

***Index Terms*—Functional Neural Network, Curse of Smoothness, Functional Data Analysis**

I. Introduction

In recent years, continuously streamed data has become pervasive; human voice and electrical signals such as EEG are notable examples. These types of data are inherently continuous, so training a parametric model often requires a large number of parameters, which can lead to the *curse of dimensionality*. To address this issue, researchers in signal processing have developed various feature engineering methods to carefully select and extract relevant features. Classic feature extraction methods include techniques such as the Fourier transform and the wavelet transform. These approaches often treat time-series data as discrete vectors, potentially losing important information about smoothness and correlated nature [1].

Functional data analysis (FDA, [2]) is a statistical framework designed for the analysis of data that are best represented as continuous functions, rather than as finite-dimensional discrete vectors. Unlike traditional approaches, FDA leverages the inherent smoothness and correlatedness of functional data, enabling more accurate modeling of complex signals. This methodology is particularly well-suited for applications involving time series, spectroscopic curves [3], or biomedical signals [4], where each data point is a realization of a continuous process. FDA encompasses a range of techniques from traditional data analysis—such as dimension reduction, regression, and classification—all specifically adapted to preserve the infinite-dimensional nature of functional observations.

With the recent success of deep neural networks (DNN) several researchers have proposed functional neural networks (FNNs) to flexibly capture the non-linear relationship between input and output data. [5] and [6] were the first to introduce FNNs that transform functional inputs into scalar hidden states, allowing standard neural networks to operate on these representations. However, this approach has been criticized for failing to preserve the functional nature in the scalar hidden states. To address this limitation, [7] and [8] proposed novel FNN architectures that maintain functional hidden states by adopting a function-on-function regression framework. They demonstrated the efficacy and superiority of fully functional neural networks in capturing complex relationships in functional data.

Currently, most components of FNNs heavily depend on those of traditional DNNs, such as element-wise activation functions and gradient-based optimization strategies. However, if the goal of an FNN is to preserve the smooth and correlated nature of functional data within its hidden states, it is crucial to consider how this property interacts with routinely used DNN components. For example, how does the element-wise activation function affect the functional nature of hidden states? Or, how does the functional nature of FNNs affect gradient-based optimization?

In this paper, we propose the *curse of smoothness*, a novel concept that explains the difficulty of training deep FNNs with stochastic gradient descent. This concept extends the well-known vanishing gradient problem, which describes the diminishing gradients in deeper layers as network depth increases. While the vanishing gradient literature emphasizes the derivative of the activation function, σ', as the primary factor, we identify function smoothness as an additional factor that affects the weight gradient norm.

II. Functional Neural Network

We begin our description of the FNN by introducing functional (generalized) linear models. Since our target problem involves functional data classification tasks, we focus on the scalar-on-function problem, where the model takes a functional input and returns a scalar output. We primarily follow the notation from [2].

Let $L^2[0,1]$ be our function space; that is, the space of square-integrable functions defined on the interval $[0,1]$. In the functional linear model, we assume the relationship between random variables $X_i \in L^2[0,1]$ and $Y_i \in \mathbb{R}$ is given by

$$Y_i = \int_0^1 \beta(t) X_i(t) dt + \alpha + \epsilon_i,$$

Submission Date: 2025.09.19. Revised: 2025.10.20. Accepted: 2025.10.20. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-24873052 and RS-2025-25433902).

The authors are with the Department of Mathematics, Korea University, 145, Anam-ro, Seongbuk-gu, Seoul, 02841, Republic of Korea (e-mail: cth127@korea.ac.kr (T.C), holy@korea.ac.kr (D.L., Corresponding author)).

Code is available at https://github.com/AIML-K/curse_of_smoothness.

where $\beta(t) \in L^2[0,1]$ is a weight function, $\alpha \in \mathbb{R}$ is a bias term, and $\epsilon_i \in \mathbb{R}$ is an *i.i.d.* error term.

We construct a functional generalized linear model for the classification of K classes by defining

$$\begin{aligned} p &= (p_1, ..., p_C) \\ &= \text{softmax}\left(\left[\int_0^1 \beta_c(t)X_i(t)dt + \alpha_c + \epsilon_{c,i}\right]_{c=1}^{C}\right), \end{aligned}$$

where Y follows a C-class categorical distribution with $\mathbb{E}[Y_i = c] = p_c$.

By extending this notation, we introduce two types of FNN structures.

a) Univariate: [8] introduced FNNs with univariate weight functions,

$$H_k^l(t) = \sigma\left(\sum_{j=1}^{J_{l-1}} \beta_{j,k}^l(t)H_j^{l-1}(t) + \alpha_k^l(t)\right), \tag{1}$$

where $H_k^l(t)$ denotes the hidden state function at the k-th node in the l-th layer, J_{l-1} is the number of nodes in $(l-1)$-th layer, and σ is a non-linear activation function. The input to the network is given by $H_{j,i}^0(t) = X_{j,i}(t)$, where $X_{j,i}(t)$ represents the j-th channel of the data point $X_i(t)$. Note that Equation (1) can be expressed as a fully connected layer with a suitable expansion. However, during training, gradient descent is applied to the basis weights rather than the weight function itself, which distinguishes FNNs from DNNs.

For the final layer, to predict scalar values, the following is used:

$$H_c^L = \sum_{j=1}^{J_{L-1}} \int_0^1 \beta_{j,c}^L(t)H_j^{L-1}(t)dt + \alpha_c^L, \tag{2}$$

and the softmax function is applied to $H^L = \left(H_1^L, ..., H_c^L\right)$. [2] refer to this as a *concurrent model*, where each element of the weight function $\beta(t)$ focuses only on t-th element of the input and hidden functions.

b) Bivariate: [7] introduced FNNs with bivariate weight functions,

$$H_k^l(t) = \sigma\left(\sum_{j=1}^{J_{l-1}} \int_0^1 \beta_{j,k}^l(s,t)H_j^{l-1}(s)ds + \alpha_k^l(t)\right).$$

We observe that the weight function $\beta_{j,k}^l(s,t) \in L^2[0,1] \times L^2[0,1]$ is bivariate, and, unlike Equation (1), the model integrates the hidden states with respect to the input domain. [7] utilizes the same univariate scheme for the final layer (Equation (2)) to produce scalar-valued predictions.

[2] refer to this as a *total model*, where $\beta(s,t)$ determines the impact of the input at time s to the output at time t. [2] and [8] also note that the bivariate scheme introduces additional complexity. As a result, the model becomes more flexible but also more difficult to optimize.

To represent the continuous and smooth weight and bias functions, β and α, both univariate and bivariate models utilize basis expansion. In other words, they represent $\beta(t) = \sum_{b=1}^{B} w_b\phi_b(t)$, where ϕ_b's are basis functions, such as the Fourier basis or Legendre basis, and w_b is the corresponding weight. In the bivariate case, $\beta(s,t) = \sum_{b=1}^{B}\sum_{c=1}^{B} w_{b,c}\psi_b(s)\phi_c(t)$. With this formulation, basis functions ψ and ϕ are treated as fixed hyperparameters, and the optimization is reduced to finding the finite set of weights w.

In standard DNNs, each hidden state H_k^l is a scalar value, whereas in FNNs, each state represents a function whose smoothness depends on the predefined bases, ϕ and ψ. While this design may be more compatible to represent the functional nature of the data, it also introduces a structural constraint: gradients must flow through smooth hidden functions and weight functions. In the next section, we show that this additional smoothness constraint can aggregate the classical vanishing gradient problem.

III. Curse of Smoothness

The vanishing gradient problem occurs when gradients used to train deep neural networks become extremely small as they are backpropagated through many layers. Since weight updates in gradient-based optimization depend on these gradients, very small values cause earlier layers (closer to the input) to learn very slowly or stop learning altogether.

Formally, for an L-layer neural network with hidden state $h_l = \sigma(W_{l-1}h_{l-1})$, where σ is an activation function, the vanishing gradient occurs for W_{l-1} if $\left\|\frac{\partial \mathcal{L}}{\partial W_{l-1}}\right\| \approx 0$, for a loss function $\mathcal{L}$ [9]. By the chain rule, $\frac{\partial \mathcal{L}}{\partial W_{l-1}} = \frac{\partial \mathcal{L}}{\partial h_L} \cdot \frac{\partial h_L}{\partial h_{L-1}} \cdots \frac{\partial h_l}{\partial W_{l-1}}$, so in deep networks the gradient can decay exponentially if each term $\left\|\frac{\partial h_l}{\partial h_{l-1}}\right\|$ is small. Moreover, since $\frac{\partial h_l}{\partial h_{l-1}} = \text{diag}\left(\sigma'(W_{l-1}h_{l-1})\right)W_{l-1}$, the vanishing gradient is especially common when using activation functions such as the sigmoid or hyperbolic tangent, whose derivatives are mostly small.

In this section, we theoretically show that not only the activation function but also the smoothness of hidden states and weight functions can cause the vanishing gradient problem. Since FNNs are designed to enforce a smooth form on both hidden states and weights, this issue becomes an inherent structural limitation of FNNs.

For simplicity, let f be a univariate FNN with one hidden dimension. Then the following theorem holds.

Theorem III.1. *For an l-th layer hidden state $H^l(t)$, let $l-1$-th weight function be decomposed as $\beta^{l-1}(t) = \sum_{b=1}^{B} w_b\phi_b(t)$. If H^{l-1} and ϕ_b are l_1 and l_2-Lipschitz with respect to t, and H^{l-1} and ϕ_b are well-normalized that they are zero at some point, then $\left\|\frac{\partial H^l}{\partial w_b}\right\|_1 \leq \sqrt{\frac{1}{5}}\left\|\sigma'(\beta^{l-1}(t)H^{l-1}(t))\right\|_2 l_1 l_2$.*

Proof. As $\frac{\partial H^l(t)}{\partial w_b} = \sigma'(\beta^{l-1}(t)H^{l-1}(t))H^{l-1}(t)\phi_b(t)$,

$$\begin{aligned} \left\|\frac{\partial H^l}{\partial w_b}\right\|_1 &= \left\|\sigma'(\beta^{l-1}(t)H^{l-1}(t))H^{l-1}(t)\phi_b(t)\right\|_1 \\ &\leq \left\|\sigma'(\beta^{l-1}(t)H^{l-1}(t))\right\|_2 \cdot \left\|H^{l-1}(t)\right\|_4 \cdot \left\|\phi_b(t)\right\|_4, \end{aligned}$$

by Generalized Hölder's Inequality. If a function g satisfies $g(t') = 0$ and is L-Lipschitz, then $|g(t)| = |g(t) - g(t')| \leq L|t - t'|$, and $\|t - t'\|_4 \leq \left(\frac{1}{5}\right)^{\frac{1}{4}}, \quad \forall t' \in [0, 1]$. So, $\left\|H^{l-1}\right\|_4 \leq \left(\frac{1}{5}\right)^{\frac{1}{4}} \cdot l_1$ and $\|\phi_b\|_4 \leq \left(\frac{1}{5}\right)^{\frac{1}{4}} \cdot l_2$, so the result holds. □

The well-normalized hidden state condition practically holds, since several normalization techniques (including layer normalization and batch normalization) have become essential in practical neural network training.

Note that this result can be naturally extended to the case of multi-dimensional case, since the Lipschitz-bound argument holds component-wise and aggregates in the same manner. Also, since the Frobenius norm of a matrix is bounded by the sum of the norms of its column vectors, we can bound the gradient norm with respect to the entire weight by summing the upper bounds of each individual weight.

In a standard DNN (with one hidden dimension), the hidden state is scalar, $H^l = \sigma(wH^{l-1}) \in \mathbb{R}$, so that $\frac{\partial H^l}{\partial w} = \sigma'(wH^{l-1})H^{l-1}$, and $\left|\frac{\partial H^l}{\partial w}\right| = \left|\sigma'(wH^{l-1})\right| \cdot \left|H^{l-1}\right|$. Importantly, H^{l-1} in DNN does not require any structural assumption. By contrast, in FNNs the gradient is additionally constrained by the Lipschitz constants l_1 and l_2, as shown in Theorem III.1, which arise from the functional smoothness of $H^{l-1}(t)$ and the basis functions $\phi_b(t)$. This highlights how smoothness itself can further suppress gradient magnitudes in FNNs. When this effect combines with the vanishing gradient problem of standard neural networks as depth increases (σ' factor), the problem becomes even more severe. We refer to this compounded difficulty as **the curse of smoothness**, an inherent structural limitation of functional neural networks.

IV. Experiments

To empirically validate the curse of smoothness, we investigate how training functional neural networks (FNNs) becomes increasingly difficult as network depth grows. Specifically, our experiments are designed to examine three questions: (1) How much harder is it to train deep FNNs compared to standard DNNs? (2) To what extent is this difficulty explained by the vanishing gradient problem? (3) Does the smoothness restriction in FNNs exacerbate the vanishing gradient effect, as predicted in Theorem III.1?

a) Datasets: For comparison, we use three signal datasets, including **Growth**, **Tecator**, and **EEG** datasets.

The **Growth** dataset [10] consists of 93 growth curves for boys and girls between the ages of 1 and 18 years, with the biological sex of each subject as the target variable. As a result, it becomes a binary classification dataset.

The **Tecator** dataset [11] contains 215 meat samples, each represented by a near-infrared absorbance spectrum (850–1050 nm) and corresponding measurements of moisture, fat, and protein content. Following [8], the primary task is to binary classify whether the fat content exceeds 20% based on the spectral data.

The **EEG** dataset is for the BCI Competition IV Dataset 2A. It consists of brain signals recorded from nine participants as they imagine movements of their left hand, right hand, both feet, and tongue. Each participant repeats this procedure 72 times for each class, resulting in a total of 2,592 data points. Unlike the other datasets, the EEG dataset includes 25 channels: 22 EEG signals and 3 EOG (electrooculography) signals.

We import the Growth and Tecator datasets from the *scikit-fda* package [12], while the EEG dataset is available at https://www.bbci.de/competition/iv/.

b) Models: We control three model-related variables: (1) *DNN vs. Univaiate and Bivariate FNN*: we examine how the vanishing gradient occurs depending on the neural network type; (2) *number of layers*: since gradients tend to vanish in deeper structures, we test networks with $L \in \{2, 4, 6\}$ layers; (3) *activation function*: we compare the sigmoid and ReLU functions, noting that the ReLU was introduced to mitigate the vanishing gradient problem in standard DNNs [13]. For both DNN and FNN, we set the hidden dimension to 256, and for FNN, we use 5 Fourier basis functions.

c) Training Details: For each dataset, we implement five-fold cross-validation and use 40% of the training set as a validation set. We use AdamW optimizer [14] with learning rate within $\{0.1, 0.05, 0.01, 0.005\}$ for 100 epochs. We select the optimal learning rate based on validation set accuracy and report the mean and standard deviation of the test set accuracy. We implement the experiments using PyTorch [15] framework.

V. Results

a) Main results: The results are summarized in Table I. In the two-layer case, FNN performs comparably to DNN on the Growth and Tecator datasets, while outperforming DNN on the EEG dataset. This gap is noteworthy given that DNN for EEG dataset consists of more than 2 million parameters, whereas FNN has only 38k parameters. Specifically, DNN achieves 52%–55% accuracy, while FNN reaches 68%–69%. These results highlight the power of *appropriately* modeling functional data with a functional model to achieve better efficiency.

However, the performance of FNN significantly deteriorates as the model becomes deeper. Although DNN also suffers from this phenomenon, its performance decreases by at most 8%–9%, whereas FNN performance drops by as much as 45%. This tendency becomes even more pronounced when the FNN structure is bivariate, except in the Growth-Sigmoid case. These results imply that optimizing FNNs becomes increasingly difficult as more layers are added.

Note that the ReLU activation function was introduced to address the vanishing gradient problem of Sigmoid-like activation functions. For DNN, ReLU appears to successfully resolve this issue in the EEG dataset, while yielding similar performance to Sigmoid in the other datasets. In contrast, for FNN, the performance drop persists not only with Sigmoid but also with ReLU. Although ReLU alleviates the drop in some cases, the results indicate that adopting ReLU alone cannot solve the complex optimization problem in FNN.

b) Effect of Smoothness: To investigate the results in depth, we examine the effect of smoothness and derivative of activation on the gradient. We compare four types of models:

TABLE I
MEAN AND STANDARD DEVIATION OF TEST SET ACCURACY SCORES FOR FIVE-FOLD CROSS-VALIDATION

Dataset	Model	Sigmoid				ReLU			
		2 layer	4 layer	6 layer	Δ Acc.	2 layer	4 layer	6 layer	Δ Acc.
Growth	DNN	95.67% (1.63)	94.62% (1.63)	96.72% (1.33)	+1.05	96.72% (0.0)	95.67% (0.0)	94.62% (0.0)	-2.10
	U-FNN	96.72% (1.63)	81.87% (4.52)	51.52% (6.18)	-45.20	96.72% (1.63)	93.56% (5.33)	90.41% (2.50)	-6.31
	B-FNN	95.67% (1.33)	76.55% (7.48)	80.70% (6.18)	-14.97	95.67% (1.33)	86.20% (2.50)	79.71% (4.52)	-15.96
Tecator	DNN	93.95% (0.71)	94.88% (1.69)	93.95% (1.30)	0.0	93.95% (1.59)	91.16% (1.92)	92.56% (2.49)	-1.39
	U-FNN	88.37% (3.71)	66.05% (4.14)	64.19% (4.43)	-24.18	93.95% (0.92)	86.51% (6.61)	78.14% (2.98)	-15.81
	B-FNN	87.44% (2.52)	68.84% (6.28)	65.58% (5.14)	-21.86	93.02% (1.42)	83.72% (5.83)	60.93% (4.67)	-32.09
EEG	DNN	55.44% (1.74)	53.63% (0.96)	46.72% (0.88)	-8.72	52.24% (1.18)	53.70% (1.37)	53.62% (1.56)	+1.38
	U-FNN	69.98% (2.11)	70.48% (2.26)	65.97% (2.23)	-4.01	69.99% (1.52)	65.51% (1.33)	65.58% (2.02)	-4.41
	B-FNN	69.68% (1.06)	66.09% (1.77)	29.82% (6.12)	-39.86	68.21% (1.49)	63.93% (2.35)	52.35% (4.70)	-15.86

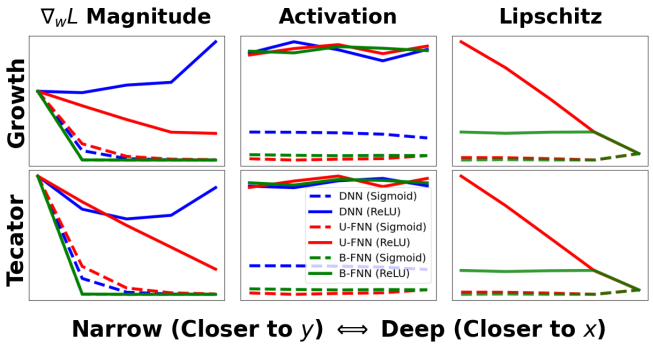

Fig. 1. (Left) Magnitude of gradient, (Middle) degree of activation, and (Right) Lipschitz constant as an inverse smoothness for each layer for (Upper) Growth and (Lower) Tecator datasets. The function smoothness results in the small gradient magnitude of the deeper layer, *the curse of smoothness*.

DNN with Sigmoid, DNN with ReLU, FNN with Sigmoid, and FNN with ReLU activation functions. We use six-layer models to verify the layer-wise interactions. Since the gradient norm varies greatly with the number of parameters, we measure the mean absolute derivative with respect to each weight, i.e., $\frac{1}{|f_l|}\sum_{w\in f_l}\left|\frac{\partial \mathcal{L}}{\partial w}\right|$ for each layer, as the magnitude of $\nabla_w \mathcal{L}$. We then normalize by the last-layer magnitude to observe the overall trend. In addition, we compute the Lipschitz constant of the hidden state function $H^{l-1}(t)$ as a measure of (inverse) function smoothness, and $\left\|\sigma'\left(\beta^{l-1}(t)H^{l-1}(t)\right)\right\|$ as the degree of activation for each layer. We explicitly compute the Lipschitz constant elementwisely. Note that, as we use the same pre-defined basis for all layers, ϕ_bs are all same. After we compute them for each data point of Growth and Tecator datasets, we report the average value. The results are shown in Figure 1.

First, the magnitude of $\nabla_w \mathcal{L}$ for Sigmoid function reveals the well-known vanishing gradient problem in both DNN and FNN [9]. The degree of activation further confirms this: since the activation degree of Sigmoid is low, the effect accumulates as the layers deepen, leading to vanishing gradients. As a result, the gradient in ReLU-DNNs recovers as the activation degree increases. In contrast, the gradient in ReLU-FNNs still decays in deeper layers, while the activation degree remains as high as that in ReLU-DNN.

On the other hand, we observe that the Lipschitz constant of the hidden state function decreases significantly in deeper layers of ReLU-FNNs. This tendency is especially severe in Bivariate FNN. We suspect that multiplying two smooth weight functions results in this smoothness. As a result, the decay in the gradient magnitude of the bivariate ReLU-FNN follows a pattern similar to that of the bivariate Sigmoid-FNN.

In summary, the hidden states of earlier layers become *too smooth*, which leads to the vanishing gradient phenomenon. This makes the optimization of deep FNNs extremely difficult, resulting in poor performance in Table I, despite growing model capacity. We can empirically verify the *curse of smoothness*, the causal relationship proposed in Theorem III.1, a phenomenon that cannot be explained by activation degree alone.

VI. CONCLUSION

We identify the *curse of smoothness*, a key phenomenon that hinders effective and robust optimization of FNNs. We theoretically show that the smoothness of functional inputs bounds the norm of the weight gradient. Since the functional form is an inherent structural component of FNNs, this smoothness becomes a fundamental limitation, especially when combined with the vanishing gradient problem. Empirically, we demonstrate that FNN optimization becomes increasingly difficult as the model depth grows, even with ReLU activation functions. In particular, we verify that gradients vanish in deeper layers of ReLU-FNNs as the hidden state functions become smoother. This phenomenon introduces a new smoothness factor beyond the activation degree, which alone cannot explain the vanishing gradient problem in ReLU-equipped FNN.

REFERENCES

[1] T. Kärnä and A. Lendasse, "Comparison of fda based time series prediction methods," in *European Symposium on Time Series Prediction*, 2007, p. 77.

[2] J. O. Ramsay and B. W. Silverman, *Applied functional data analysis: methods and case studies*. Springer, 2002.

[3] W. Saeys, B. De Ketelaere, and P. Darius, "Potential applications of functional data analysis in chemometrics," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 22, no. 5, pp. 335–344, 2008.

[4] Y. Zhang, C. Wang, F. Wu, K. Huang, L. Yang, and L. Ji, "Prediction of working memory ability based on eeg by functional data analysis," *Journal of neuroscience methods*, vol. 333, p. 108552, 2020.

[5] B. Conan-Guez, "Multi-layer perceptrons for functional data analysis," *HAL*, vol. 2002, 2002.

[6] F. Rossi, B. Conan-Guez, and F. Fleuret, "Functional data analysis with multi layer perceptrons," in *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, vol. 3. IEEE, 2002, pp. 2843–2848.

[7] A. R. Rao and M. Reimherr, "Nonlinear functional modeling using neural networks," *Journal of Computational and Graphical Statistics*, vol. 32, no. 4, pp. 1248–1257, 2023.

[8] F. Heinrichs, M. Heim, and C. Weber, "Functional neural networks: Shift invariant models for functional data with applications to eeg classification," in *International Conference on Machine Learning*. PMLR, 2023, pp. 12 866–12 881.

[9] B. Hanin, "Which neural net architectures give rise to exploding and vanishing gradients?" *Advances in neural information processing systems*, vol. 31, 2018.

[10] R. D. Tuddenham, "Physical growth of california boys and girls from birth to eighteen years," *University of California publications in child development*, vol. 1, no. 2, 1954.

[11] F. Ferraty, *Nonparametric functional data analysis*. Springer, 2006.

[12] C. Ramos-Carreño, J. L. Torrecilla, M. Carbajo Berrocal, P. Marcos Manchón, and A. Suárez, "scikit-fda: A Python Package for Functional Data Analysis," *Journal of Statistical Software*, vol. 109, no. 2, pp. 1–37, May 2024. [Online]. Available: https://www.jstatsoft.org/article/view/v109i02

[13] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[14] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[15] A. Paszke, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.