`WFNet`: Sinusoidal Weights for Fourier Expressibility

Noah Schliesman

Draft: 10/06/2024

Abstract

Inspired by Fourier decomposition and the potential benefits of wave-based weights in neural networks, we propose `WFNet`, a neural network architecture utilizing sinusoidal weights for digit classification on the MNIST dataset. While standard neural networks approximate functions via hyperplane projections, we investigate whether sinusoidal weight functions can outperform traditional methods. Our results indicate that the accuracy plateaued at 97.4%, with no further improvements achieved. Despite the promising theoretical framework, gradient analysis highlighted potential limitations in training stability and learning dynamics, particularly due to the complex oscillatory nature of the weight parameterization.

Neural Networks and Fourier Series

As neural networks serve as powerful function approximators, it is logical to revisit the theory of Fourier series. For computational feasibility, we consider the discrete case:

$$S_N[n] = \sum_{k=-\infty}^{\infty} s[k] \exp\left(i2\pi \frac{k}{N} n\right) \tag{1}$$

where:

- $S_N[n]$ is the discrete signal,
- $s[k]$ is the k-th Fourier component,
- n is the discrete time index,
- $\alpha = 2\pi \frac{k}{N}$ is the angular frequency at k,
- i is the imaginary unit.

The concept of a Fourier activation can be considered in terms of function approximation. Ngom and Marin [1] demonstrated that Fourier networks can be effective in modeling and solving partial differential equations (PDEs) with periodic boundary conditions. Similarly, Uteluliyeva et al. [2] evaluated Fourier networks and noted that they are inferior to the simpler sigmoid and ReLU counterparts for certain tasks. More recently, Mehrabian et al. [3] utilized a Fourier-Kolmogorov-Arnold network to provide continuous and resolution-independent approximations, known as implicit neural representations (INRs). Furthermore, Sitzmann et al. [4] proposed a sinusoidal activation framework

to represent images, wavefields, video, sound, and other data types, which can also solve boundary problems. Clearly, extensive research encompasses functional approximation and differential analysis. The notion of waves as universal approximators holds merit and forms the basis for modern physics.

Wave Functions in Neural Networks

It may be advantageous to consider the weights in a neural network as wave functions. Such a system benefits from a bound of $[-A, A]$. We define the following learnable parameters:

- A: amplitude of oscillation,
- f: frequency of oscillation,
- Φ: phase shift of oscillation,
- b: vertical offset.

For a weight $w \in W$, each weight becomes:

$$w = A\sin(fx + \Phi) + b \tag{2}$$

conditioned on the layer input X. The output of a single neuron y is computed as:

$$z_j = \sum_{i=1}^{n} w_i x_i + B_j = \sum_{i=1}^{n} \left(A\sin(fx_i + \Phi) + b\right) x_i + B_j \tag{3}$$

As revealed in Ba et al.'s work on layer normalization [5], statistical regulation stabilizes the hidden state dynamics. Let:

$$\mu = \frac{1}{m}\sum_{j=1}^{m} z_j \tag{4}$$

$$\sigma^2 = \frac{1}{m}\sum_{j=1}^{m}(z_j - \mu)^2 \tag{5}$$

$$\hat{z_j} = \frac{z_j - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{6}$$

For numerical stability, we use a constant $\epsilon \to 0$, and with trainable parameters γ and β, we normalize the layer output:

$$y_j = \gamma\hat{z_j} + \beta_j \tag{7}$$

Gradient Analysis

To ensure stability, it is informative to perform gradient analysis:

$$\frac{\partial y}{\partial A} = \frac{\gamma x \sin(fx + \Phi)}{\sqrt{\sigma^2 + \epsilon}} \tag{8}$$

$$\frac{\partial y}{\partial f} = \frac{-Ax^2 \cos(fx + \Phi)}{\sqrt{\sigma^2 + \epsilon}} \tag{9}$$

$$\frac{\partial y}{\partial \Phi} = \frac{-Ax \cos(fx + \Phi)}{\sqrt{\sigma^2 + \epsilon}} \tag{10}$$

$$\frac{\partial y}{\partial b} = \frac{\gamma x}{\sqrt{\sigma^2 + \epsilon}} \tag{11}$$

$$\frac{\partial y}{\partial \gamma} = \frac{z_j}{\sqrt{\sigma^2 + \epsilon}} \tag{12}$$

$$\frac{\partial y}{\partial y} = \frac{\beta - \mu + x\left[A \sin(fx + \Phi) + b\right]}{\sqrt{\sigma^2 + \epsilon}} \tag{13}$$

$$\frac{\partial y}{\partial \beta} = 1 \tag{14}$$

Initially, the frequency gradient poses concerns. In the event that this architecture fails, we will investigate potential issues with $\frac{\partial y}{\partial f}$.

Activation Function Testing

Choosing an appropriate activation function is non-trivial. As a baseline, we test:

- (a) Sigmoid: $y_{\text{act}} = \sigma(y_i)$,
- (b) ReLU: $y_{\text{act}} = \text{ReLU}(y_i)$.

This concludes the basic architecture of our parameterizable wave function weight network, which we refer to as "WFNet." Before proceeding with further analysis, it is advantageous to test on a toy dataset. To this end, we create a network that employs the WFNet architecture to classify digits in the MNIST dataset.

Analysis of Sinusoidal Weights via Hyperbolic Attractors

In deep recurrent architectures, the phenomena of vanishing and exploding gradients pose significant challenges for training [6]. This issue can be analyzed through the lens of dynamical systems, where the recurrence relations in neural networks form discrete-time dynamical systems. By examining the system's hyperbolic attractors, we can understand gradients during backpropagation. We extend this analysis to networks with sinusoidal weights, exploring how the sinusoidal parameterization affects gradient flow and training dynamics.

Dynamical Systems Perspective

Consider a recurrent neural network (RNN) where the hidden state h_t at time step t is defined as:

$$h_t = \phi(Wh_{t-1} + Ux_t + b) \tag{15}$$

where:

- $h_t \in \mathbb{R}^n$ is the hidden state vector,
- $x_t \in \mathbb{R}^m$ is the input vector,
- $W \in \mathbb{R}^{n \times n}$ is the recurrent weight matrix,
- $U \in \mathbb{R}^{n \times m}$ is the input weight matrix,
- $b \in \mathbb{R}^n$ is the bias vector,
- $\phi(\cdot)$ is an activation function (e.g., tanh, ReLU).

The evolution of h_t can be characterized by fixed points and their stability, which are determined by the eigenvalues of the Jacobian $J_t = \frac{\partial h_t}{\partial h_{t-1}}$.

Sinusoidal Weights in Recurrence Relations

In our sinusoidal weight framework, the weights are parameterized as:

$$W_{ij} = A_{ij} \sin(f_{ij} h_{t-1,j} + \Phi_{ij}) + b_{ij} \tag{16}$$

Substituting W into the recurrence relation:

$$h_t = \phi\left(\sum_{j=1}^{n} \left[A_{ij} \sin(f_{ij} h_{t-1,j} + \Phi_{ij}) + b_{ij}\right] h_{t-1,j} + Ux_t + b\right) \tag{17}$$

This introduces higher-order nonlinearity and dependence on h_{t-1}, complicating the dynamical analysis.

Jacobian Matrix and Gradient Propagation

The Jacobian matrix J_t is crucial for understanding gradient flow during backpropagation:

$$J_t = \frac{\partial h_t}{\partial h_{t-1}} = \phi'(z_t) \left(\frac{\partial z_t}{\partial h_{t-1}} \right) \tag{18}$$

where $z_t = Wh_{t-1} + Ux_t + b$ is the pre-activation vector, and $\phi'(z_t)$ is a diagonal matrix of activation function derivatives.

Computing the partial derivative $\frac{\partial z_t}{\partial h_{t-1}}$:

$$\frac{\partial z_{t,i}}{\partial h_{t-1,k}} = \frac{\partial}{\partial h_{t-1,k}} \left(\sum_{j=1}^{n} W_{ij} h_{t-1,j} \right) = W_{ik} + \sum_{j=1}^{n} h_{t-1,j} \frac{\partial W_{ij}}{\partial h_{t-1,k}} \tag{19}$$

Since W_{ij} depends on $h_{t-1,j}$, we need to compute $\frac{\partial W_{ij}}{\partial h_{t-1,k}}$:

$$\frac{\partial W_{ij}}{\partial h_{t-1,k}} = \delta_{jk} \left[A_{ij} f_{ij} \cos(f_{ij} h_{t-1,j} + \Phi_{ij}) \right] \tag{20}$$

where δ_{jk} is the Kronecker delta. Substituting back into $\frac{\partial z_{t,i}}{\partial h_{t-1,k}}$:

$$\frac{\partial z_{t,i}}{\partial h_{t-1,k}} = W_{ik} + h_{t-1,k} \left[A_{ik} f_{ik} \cos(f_{ik} h_{t-1,k} + \Phi_{ik}) \right] \tag{21}$$

Eigenvalue Analysis

The behavior of gradients is influenced by the eigenvalues of J_t. If the spectral radius $\rho(J_t)$ is:

- Less than 1: Gradients tend to vanish,
- Greater than 1: Gradients tend to explode.

For sinusoidal weights, W_{ij} and $\frac{\partial W_{ij}}{\partial h_{t-1,k}}$ involve sinusoidal and cosinusoidal terms, which can oscillate between $-A_{ij}$ and A_{ij}.

Hyperbolic Attractors and Stability Analysis

Hyperbolic attractors are characterized by having no eigenvalues of modulus one, leading to exponential divergence or convergence along different directions in the state space. Linearizing the system around a fixed point h^*:

$$h^* = \phi(Wh^* + Ux + b) \tag{22}$$

the Jacobian at h^* is:

$$J^* = \phi'(z^*) \left(W + \frac{\partial W}{\partial h_{t-1}} h^* \right) \tag{23}$$

The presence of $\frac{\partial W}{\partial h_{t-1}}$ introduces additional terms that can significantly affect the eigenvalues of J^*. Specifically, the term involving $\cos(f_{ij} h_{t-1,j} + \Phi_{ij})$ can cause the eigenvalues to fluctuate, potentially crossing the unit circle in the complex plane.

Results

The implemented model, `WFNet`, uses custom wave-based weights, where the weights dynamically vary as a sinusoidal function of the input, applied to the MNIST dataset. Despite the use of complex sinusoidal weight behavior, the model's accuracy plateaued at **97.4%**, showing no further improvement. The training loss decreased consistently, but after a certain point, the test accuracy remained stagnant. This suggests that the wave-based approach may have reached its capacity for this task or could indicate early convergence or overfitting to the training data. Fine-tuning, such as adjusting hyperparameters or exploring regularization techniques, might be required to surpass this threshold. For the code, please contact the author.

Conclusion

Analyzing sinusoidal weights through the perspective of hyperbolic attractors provides valuable insights into the training dynamics of neural networks with such weights. The oscillatory nature introduces complex dynamics that can exacerbate the vanishing and exploding gradient problem. Understanding these dynamics is crucial for designing effective training strategies and ensuring network stability.

References

1. M. Ngom and O. Marin, "Fourier Neural Networks as Function Approximators and Differential Equation Solvers," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 14, 2021, pp. 641–661. https://doi.org/10.1002/sam.11331

2. U. Uteluliyeva et al., "Fourier Neural Networks: A Comparative Study," *Intelligent Data Analysis*, vol. 24, no. 5, 2020, pp. 1107–1120.

3. A. Mehrabian et al., "Implicit Neural Representations with Fourier Kolmogorov-Arnold Networks," arXiv preprint arXiv:2409.09323v2, 2024.

4. V. Sitzmann et al., "Implicit Neural Representations with Periodic Activation Functions," *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 7462–7473.

5. J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," arXiv preprint arXiv:1607.06450, 2016.

6. Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, 1994, pp. 157–166.

7. G. Klambauer et al., "Self-Normalizing Neural Networks," *Advances in Neural Information Processing Systems*, vol. 30, 2017.