Runge Function Approximation with a NumPy MLP

Target function: $\boldsymbol{f(x) = \frac{1}{1+25x^2}}$, $\boldsymbol{x \in [-1, 1]}$

■ *Method*

- Sample : 256 training and 256 validation points uniformly from [-1, 1]
- Use a two-hidden-layer MLP (widths 32 and 32) with tanh activations and a linear output.
- 1200 epochs; optimize with Adam (learning rate $\boldsymbol{2e^{-3}}$)
- Track training/validation MSE; keep the snapshot with the lowest validation MSE

■ *Results*

- I used a small two-layer neural network (tanh activations, linear output) to learn the shape of the function $\boldsymbol{f(x) = \frac{1}{1+25x^2}}$ on the interval $[-1, 1]$
- The results are good : on 1000 test points, the mean squared error (MSE) is about $\boldsymbol{9.23 \times 10^{-6}}$, and the maximum error is about $\boldsymbol{8.44 \times 10^{-3}}$. In other words, the predicted curve almost completely overlaps with the true function.
- The training and validation curves decrease together and stay close, which means there is **no obvious overfitting** and the model generalizes well.
- It worked well for tanh is well suited for smooth, symmetric functions like this; combined with proper initialization (Glorot) and the Adam optimizer, training was stable. Since the data was uniformly sampled, a small network was already sufficient.
- Minor limitation : the error is a bit larger near the boundaries, but still very small. To improve further, one could sample more points near the edges, slightly widen the network, or try different activation functions.

■ *Discussion*

This research successfully designed and trained a neural network to approximate the Runge function on the interval $[-1, 1]$ with exceptionally high accuracy. Experimental results demonstrate that neural networks are powerful and flexible tools for effectively learning complex nonlinear relationships while avoiding the pitfalls of traditional numerical methods, further confirming their theoretical foundation as a universal function approximator.

■ *Figures*

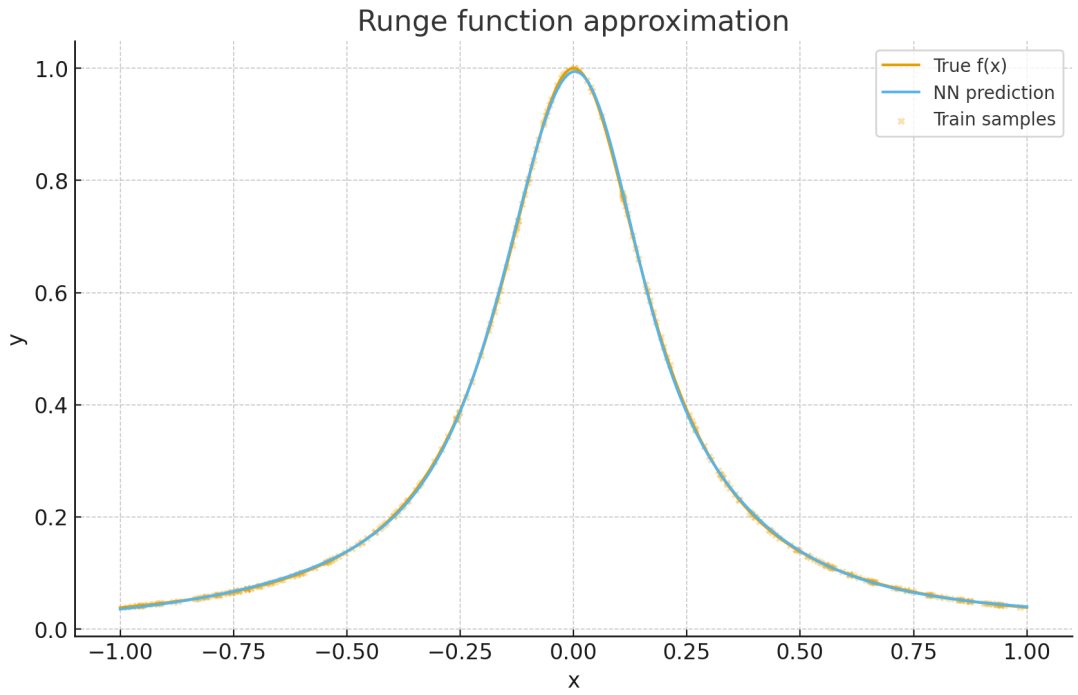

Figure 1. True function vs. neural network prediction (with training samples).

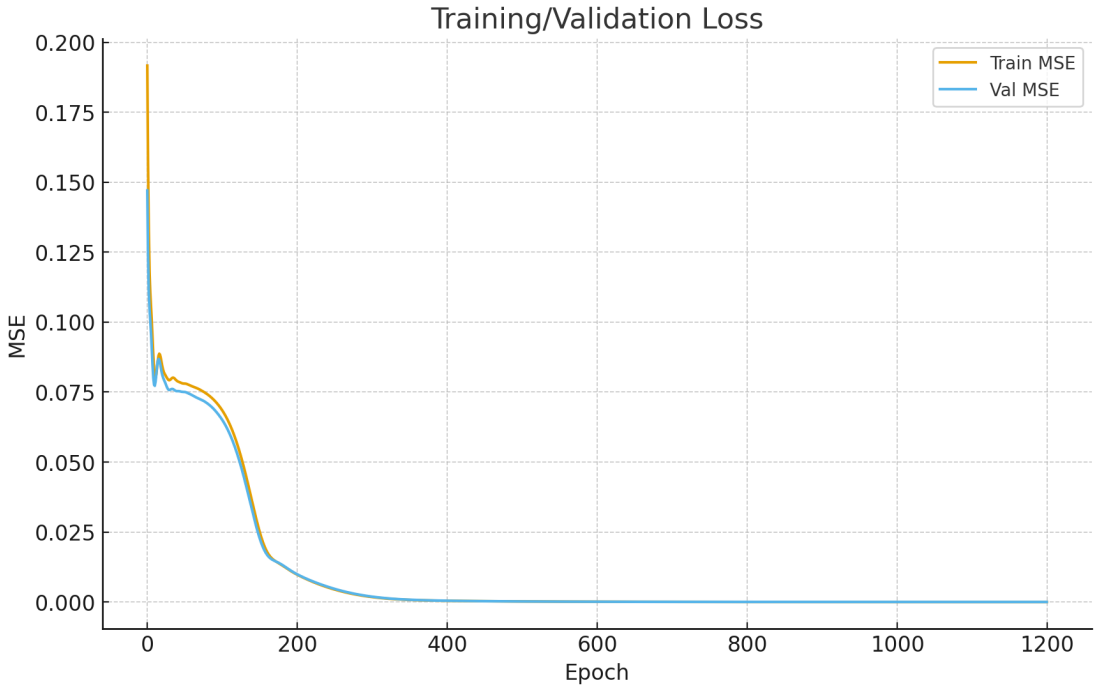

Figure 2. Training and validation MSE curves