# An Exploration of Neural Network Expressivity and Approximation Dynamics

Dylan Hu

May 11, 2023

## Contents



## 1 Introduction

In this report, we explore the expressivity and approximation dynamics of neural networks. In particular, we aim to investigate the relationship between a handful of specific architectural decisions when building multi-layer perceptrons and relate these findings to the theoretical bases of expressivity and universal approximation. Furthermore, we explore the effects of additional techniques, such as positional encoding, to facilitate more efficient approximations of complex functions.

The expressivity of a neural network refers to the diversity and complexity of functions it can represent. This complexity is primarily determined by the architecture of the network, especially its depth and width. The depth of a neural network, represented by the number of layers it possesses, and the width, indicated by the number of neurons in each layer, play crucial roles in determining the network's expressive power. By manipulating these parameters, one can significantly alter the kind of functions a network can efficiently approximate.

One particularly fascinating notion with these types of neural networks is their universal approximation property, as posited in proofs by Cybenko and

Hanin and Sellke, among others. According to Cybenko, a sufficiently wide feed-forward neural network with a single hidden layer can approximate any continuous function to an arbitrary degree of accuracy, given that the network's activation function is a non-constant, bounded, and monotonically-increasing continuous function. This theorem posits that neural networks are universally capable of learning and representing a broad array of functions, given enough neurons in the hidden layer. In practice, however, we often find that deeper networks with multiple layers can learn more efficiently and generalize better to unseen data.

In this project, we put these theoretical assertions to the practical test. We explore how varying the depth and width of a neural network affects its ability to approximate given functions. We specifically focus on a class of functions that are generated using random Fourier series, which are increasingly oscillatory, mirroring real-world functions that neural networks often grapple with. By observing how these networks learn to fit these functions over time, we gain insights into the dynamics of the training process.

In addition, we touch upon the concept of positional encoding, a technique often used in various types of applications including language modeling, where it helps to provide the model with a sense of order of the input data. Positional encoding is especially beneficial for high-frequency fits, enabling the model to handle complex, oscillatory functions more effectively. By incorporating positional encoding in our neural networks, we demonstrate enhanced approximation efficiency.

As we delve into these intriguing aspects of neural networks, we keep our focus on understanding the interplay of theory and practice in neural network design and training. Through this project, we strive to bridge the gap between what is theoretically possible and what is practically achievable, shedding light on how to more efficiently and effectively leverage the power of neural networks in solving complex problems.

## 2 Method

We consider the task of approximating a function given by a set of values on a regular grid of values $0 = x_0, \ldots, x_M = 1$. As such, the function to approximate takes the form $f : [0, 1] \to \mathbb{R}$, and we refer to our model's approximation as $g : [0, 1] \to \mathbb{R}$. We consider the following class of functions:

$$f(x) = \sum_{i=1}^{N} a_i \sin(2\pi i x) + b_i \cos(2\pi i x) \tag{1}$$

where $a_i, b_i$ are randomly chosen. This class of functions is generated using random Fourier series, which are increasingly oscillatory and mirror real-world functions that neural networks often grapple with. We use this class of functions to explore the expressivity and approximation dynamics of neural networks.

We implement our experiments in PyTorch 2. Unless otherwise specified, we use the Adam optimizer with a learning rate of 0.001 and train for 1000 epochs.

## 2.1 Neural Network Architecture

This project utilizes a multi-layer perceptron (MLP), a type of feed-forward deep neural network. The MLP is composed of multiple layers, each containing a number of neurons or nodes, and utilizes an activation function for introducing non-linearities.

The network is characterized by a depth, $N$, indicating the total number of layers, and widths, represented by $d_0, d_1, ..., d_N$. The conditions $d_0 = d_N = 1$ are imposed to denote a single neuron in the input and output layers, a typical configuration for function approximation tasks. The widths of the hidden layers, $d_1, ..., d_{N-1}$, are varied throughout the project.

The MLP follows a fully-connected architecture, where each neuron in a layer is connected to all neurons in the following layer. Various activation functions, including sigmoid, ELU (Exponential Linear Unit), and ReLU (Rectified Linear Unit), are applied to add non-linear transformations at each neuron.

## 2.2 Optimization Objective

The parameters of the network, including the weights and biases (collectively referred to as $w$), are learned during the training process. The goal is to minimize the difference between the network's predictions and the actual function values by iteratively adjusting these parameters through gradient descent in order to minimize a mean squared error (MSE) loss function. The loss function is defined as follows:

$$L(g) = \frac{1}{M} \sum_{j=0}^{M} (g(x_j; w) - f(x_j))^2 \tag{2}$$

## References