PDL Challenge 1: Neural Network-Based Function Upscaling: The Role of Activation Functions

SR+ITL

March 22, 2024

Abstract

Reconstructing a function from its samples is a cornerstone problem in engineering, underpinning technologies in domains ranging from digital communications to audio and image processing. Traditionally, this problem has been approached through linear reconstruction methods, such as sinc interpolation, leveraging the mathematical framework provided by the Nyquist-Shannon sampling theorem. These methods, however, often assume ideal conditions and may not adequately handle real-world complexities such as noise, non-linear distortions, or incomplete sampling.

The advent of Deep Neural Networks (DNNs) has introduced a paradigm shift in how we approach the reconstruction of functions from samples. DNNs offer a powerful toolkit for learning non-linear mappings, providing a means to generalize the traditional reconstruction operation far beyond linear methods. This generalization capability enables DNNs to effectively deal with the intricacies of real-world signals and sampling conditions, paving the way for advancements in signal processing and related fields.

In this exploration, we delve into the utilization of DNNs for the task of upscaling band-limited functions from their sparse samples. We will investigated various factors in the reconstruction process, including the network architecture, the size of the dataset, and, most crucially, the choice of activation functions. Activation functions, the heartbeats of neural networks, introduce non-linearities essential for the network's learning capability. Their characteristics significantly influence the network's ability to approximate complex functions and, by extension, to reconstruct and upscale band-limited functions accurately.

In this challenge, we aim to provide a comprehensive understanding of how these elements interact within the context of function reconstruction. This will not only shed light on the practical aspects of signal processing with deep learning but also highlight the theoretical underpinnings that make such advancements possible. As we embark on this exploratory journey, we invite students to engage with these concepts actively, fostering a deeper appreciation for the challenges and opportunities that lie at the intersection of deep learning and signal reconstruction.

1 Problem Setting

1.1 Bandlimited discrete Functions

In the realm of digital signal processing, discrete-time functions play a pivotal role, especially when it comes to the analysis and processing of digital signals. One of the most powerful tools for understanding and manipulating these functions is the Fourier Transform, which allows us to represent a time-domain signal in terms of its frequency components. Specifically, for a real-valued discrete-time function, its Fourier Transform yields a set of coefficients that encapsulate the function's frequency content.

These coefficients are complex numbers, each representing the amplitude and phase of a sinusoidal component of the original function. Given the real nature of our discrete-time functions, these Fourier coefficients exhibit specific symmetries:

- **Magnitude Symmetry:** The magnitudes of the Fourier coefficients are symmetric about the zero-frequency component. This means that the amplitude of the negative frequency components mirrors that of the positive frequency components.
- **Phase Symmetry:** The phases of the Fourier coefficients are antisymmetric around the zero-frequency component. Essentially, this implies that if a coefficient at a positive frequency has a certain phase, the corresponding negative frequency coefficient will have the opposite phase.

This symmetry arises from the fundamental properties of the Fourier Transform applied to real-valued signals and serves as a crucial concept in our exploration.

Generating Band-limited Functions To study the reconstruction of functions from their samples, we focus on band-limited functions. A band-limited function is one whose Fourier Transform (or spectrum) is non-zero only within a specific finite frequency range and zero elsewhere. This characteristic implies that the function does not contain arbitrarily high frequencies and, as a consequence, varies smoothly between samples.

In our challenge, we generate discrete-time functions by randomly selecting the Fourier Transform coefficients within a defined band-limit. By controlling these coefficients, we ensure that our functions are band-limited, adhering to the desired property of having a smooth variation between samples. The process involves:

Randomly picking the magnitude and phase for each Fourier coefficient within the band-limit, ensuring adherence to the magnitude and phase symmetry conditions for real-valued functions. Applying the inverse Fourier Transform to these coefficients to obtain the time-domain representation of our band-limited function. This method of generating functions allows us to create a well-behaved class of signals for experimentation. The band-limited nature of these functions is particularly advantageous for our study, as it aligns with the theoretical prerequisites for perfect reconstruction from sparse samples, according to the Nyquist-Shannon sampling theorem.

In this setting, our exploration will dissect the impact of neural network architectures, dataset sizes, and especially the choice of activation functions on the task of reconstructing these band-limited functions from their discrete samples. Through this exploration, we aim to unearth the nuances of function approximation with deep neural networks and the critical role played by activation functions in this context.

Some sample code is as follows

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(42)  # Fixing the random seed for reproducibility

# Parameters
N = 512  # Total number of points
    band_limit = 50  # Band-limit (frequency components will be non-zero only within
    this limit)

# Generate random Fourier coefficients
coefficients = np.zeros(N, dtype=np.complex)
# Since the function is real, we need to ensure the symmetry of the coefficients
coefficients[1:band_limit] = np.random.randn(band_limit-1) + 1j * np.random.randn(
    band_limit-1)  # Positive frequencies
```

```
coefficients[-(band_limit-1):] = coefficients[1:band_limit][::-1].conj()  # Negative
    frequencies, conjugate symmetry

# Inverse FFT to generate the band-limited time-domain function
time_domain_function = np.fft.ifft(coefficients).real

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(time_domain_function, label='Band-limited Function')
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.title('Band-limited Function Generated from Random Fourier Coefficients')
plt.legend()
plt.show()
```

2 Tasks Description

This challenge is divided into two main tasks, each designed to deepen your understanding of neural network-based function approximation and reconstruction. These tasks will guide you through the process of training a deep neural network (DNN) to learn the mapping between inputs and outputs represented by discrete points and sampled points from band-limited functions, respectively. By completing these tasks, you will gain insights into how DNNs interpolate and extrapolate beyond the provided data.

Let us start by describing the tasks from a high-level perspective: after that, we will describe the DNN architectures and then provide a detailed description of the tasks.

- **Task 1: A DNN Learns to Reconstruct one Band-limited Function.**
  The first task focuses on training a neural network to approximate a band-limited function defined by a set of discrete x and y points. The objective is for the DNN to memorize (or overfit) the (x, y) pairs in the dataset. Once the network is able to reproduce this mapping accurately, we wish to investigate how the network's ability to map any input to output varies as a function of the dataset size, the network depth, and the choice of the activation function.

- **Task 2: A DNN Learns to Interpolate All Band-limited Functions.**
  The second task extends the concept of function approximation to the reconstruction of band-limited functions from their sampled points. Here, we begin by choosing a subset of points in the input space—denoted as $\mathcal{X}$. A dataset is then created as follows: A random band-limited function is generated and used to create one entry in the dataset, where one entry corresponds to a vector of the function evaluated at the points in $\mathcal{X}$, and the label corresponds to the function evaluated over the entire (discrete) input space. A DNN is then trained to reproduce the whole function given a subset of samples. Once the network is able to interpolate the class of band-limited functions, we wish to investigate how the network's ability to map any input to output varies as a function of the dataset size, the network depth, and the choice of the activation function.

3 Exploring the Neural Architecture Landscape

In this challenge, both tasks are designed to probe into the intricate relationship between dataset size, neural network architectures, activation functions, and their collective impact on model performance. Despite sharing these common variables, each task is uniquely framed within its own constraints and objectives, highlighting the adaptability required in designing and optimizing neural networks for specific problems.

3.1 Architectural and Dataset Constraints

Task 1: Aimed at reconstructing a band-limited function from discrete (x, y) points, this task challenges the participants to work within a *10,000 neurons budget*. The network is constrained to a single input and a single output, simulating a straightforward, yet nuanced function approximation scenario.

Task 2: Elevates the complexity by focusing on interpolating all possible band-limited functions from a set of sampled points. Here, the neuron budget is significantly increased to *1,000,000 neurons*, accommodating a broader input space of 100 inputs and an expanded output dimension of 512. This task delves into the network's capacity for handling multi-dimensional function reconstruction.

3.2 Training Specifications

Both networks are to be trained under a mean squared error (MSE) loss function utilizing the Adam optimizer, with training proceeding until convergence. This approach ensures that the focus remains on the network's learning capabilities and the effectiveness of the architectural decisions made within the specified neuron budget constraints.

3.3 Experimental Variations

Participants are encouraged to experiment with various aspect of the DNN architecture and dataset size

- **Network Depths and Layer Sizes** Explore the effect of different numbers of layers and different numbers of neurons per layers while ensuring the total number of neurons does not exceed the allocated budget for each task.
- **Different Activation Functions:** Explore the influence of different activation functions on the network's ability to learn and generalize from the given data. Consider how nonlinearities introduced by these functions affect the interpolation and reconstruction capabilities of the network.
- **Varying Dataset Sizes:** Investigate how changes in the size of the training dataset impact model performance. This exploration should provide insights into the relationship between data availability and the network's capacity to approximate or interpolate functions accurately.

The underlying goal across these tasks is to foster a deep understanding of how various elements of neural network design—ranging from architectural choices to the subtleties of activation functions and dataset characteristics—converge to influence model outcomes. Participants are expected to critically analyze their findings, drawing conclusions that are as nuanced as the networks they design.

4 Task 1: A DNN Learns to Reconstruct one Band-limited Function

The first task focuses on training a neural network to approximate a function defined by a set of discrete x and y points. The objective is to "teach" the network to produce a specific output for a given input, based on the provided dataset.

1. **Randomly Generate a Target Function:** using the code example above, randomly generate a band-limited function with input in the range $x = 1 \dots 512$ and with 50 non-zero (complex) Fourier coefficients.

2. **Dataset Preparation:** Let M be the dataset size. Create the M entries in the dataset by choosing M values of x and inserting in the dataset the predictor/label pair (x, y) that represent the input-output mappings of the target function.

3. **Network Training:** Train a neural network using the prepared dataset. The network should learn to approximate the underlying function that maps the given x points to their corresponding y points under an L_2 loss.

4. **Experiment with the DNN Architecture:** Having fixed the dataset, repeat the training for different network architectures as in Sec. 3.

5. **Experiment with dataset size:** Observe the change in approximation accuracy and overall quality as a function of the dataset size.

6. **Experiment with evaluation points:** Let $\mathcal{X}$ be the positions x in which the function is evaluated. Observe the change in approximation accuracy and overall quality as a function $\mathcal{X}$, i.e. irregual vs irregular.

4.1 Task 2: Reconstruction of Band-limited Functions

1. **Randomly Generate a M Target Function:** Let M be the dataset size, using the code example above, randomly generate M band-limited function with input in the range $x = 1 \dots 512$ and with 50 non-zero Fourier coefficients. Let the randomly generated function at the m^{th} iteration for $m \in [1, \dots M]$ is indicated as f_m.

2. **Choose** 100 **sampling points** For the set of 512 input choose 100 indexes – for example uniformly or exponentially spaced. Call the subset of indexes $\mathcal{X}$.

3. **Dataset Preparation:** Generate a dataset as follows: for the entry $m \in [1, \dots, M]$

   - the predictor is a vector of size 100 corresponding to f_m evaluated on $\mathcal{X}$
   - the label is a vector of size 512 corresponding to f_m evaluated on $[1, \dots, 512]$.

4. **Network Training:** Use the sampled points as the training dataset for a DNN. The network's task is to learn the mapping from the sampled points back to the original band-limited function.

5. **Observation:** Upon completing the training, evaluate how the network reconstructs the band-limited function. Analyze the network's performance in terms of its ability to interpolate between the sampled points and how it generalizes the class of band-limited functions beyond the provided samples.

6. **Experiment with dataset size:** Observe the change in approximation accuracy and overall quality as a function of the dataset size.

7. **Experiment with the choice of $\mathcal{X}$:** Observe the change in approximation accuracy and overall quality as a function the choice of $\mathcal{X}$: regular or irregular.

5 Observe, Observe, Observe

In the realm of deep learning, the art of observation transcends beyond mere data analysis; it is about crafting hypotheses, conducting experiments, and drawing insights from the results. The ability to extrapolate conclusions from numerical experiments is a cornerstone of scientific inquiry and innovation within this field. As practitioners and researchers, we continuously make assumptions, test them through rigorous numerical experimentation, and refine our understanding based on the outcomes.

5.1 The Art of Hypothesis Testing

Deep learning, with its myriad of architectures and hyperparameters, presents a fertile ground for hypothesis-driven exploration. A fundamental aspect of this exploration is understanding how different factors, such as dataset size, network architecture, and activation functions, influence the learning process and the model's performance. For instance, one might hypothesize about the *effect of dataset size on the error decay rate*. Such a hypothesis could be framed as: "As the dataset size increases, the error decay rate improves, leading to better model performance."

5.2 Framework for Testing Hypotheses

To verify a hypothesis like the one above, it is essential to establish a structured framework for numerical experimentation:

1. **Define the Variables:** Clearly delineate the independent variables (e.g., dataset size, architecture depth) and the dependent variable (e.g., error decay rate).
2. **Control for Confounders:** Ensure that other factors are controlled for or kept constant to isolate the effect of the independent variable on the dependent variable.
3. **Select a Range of Conditions:** Experiment with a broad spectrum of conditions to capture diverse scenarios. For dataset size, this could mean testing with very small to very large datasets.
4. **Replicate Experiments:** Conduct multiple runs with different random initializations to account for variability in the results.
5. **Analyze Results:** Use statistical methods to analyze the data, looking for patterns or trends that support or refute the hypothesis.
6. **Draw Conclusions:** Based on the analysis, conclude whether the data supports the hypothesis and under what conditions.