Verification of a Neural Network that predicts the frequency of a given Signal

Deeksha, Soumodev

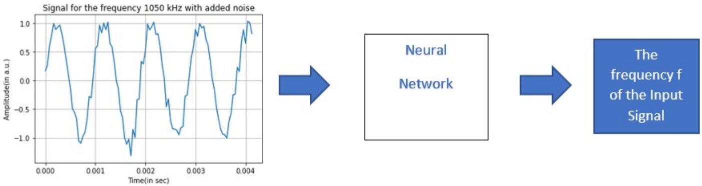
Chennai Mathematical Institute

Table of contents

Problem Statement

- We want to predict the frequency of a given sinusoidal wave using Neural Network

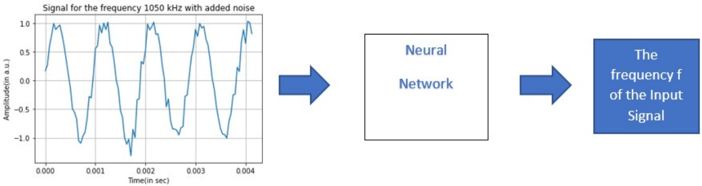
Example



$$S(t) = A.sin(2\pi f t + \varphi) + \Omega(t)$$

- What do you mean by "Signal as an input"?

Example



$$S(t) = A.sin(2\pi ft + \varphi) + \Omega(\mathrm{t})$$

- What do you mean by "Signal as an input"?
- A List of Sampling points!

Problem Statement

- We did Abstraction refinement on the Neural Network.

Problem Statement

- We did Abstraction refinement on the Neural Network.
- We want to verify some properties of the Trained Neural Network.

Motivation for using Neural Network

- Estimation of the frequency, f of a noisy signal has been one of the main problems in the field of signal processing and communications, due to its vast applications including power systems, communications, and radar.

Motivation for using Neural Network

- Estimation of the frequency, f of a noisy signal has been one of the main problems in the field of signal processing and communications, due to its vast applications including power systems, communications, and radar.
- We can use Neural Network since one of the issues with implementing theoretical techniques is that it is quite tedious to calculate every time for each input.

Motivation for using Neural Network

- It is difficult to get a mathematical identification because of the noise in the signal. NNs finds complex patterns from such inputs.

Motivation for using Neural Network

- It is difficult to get a mathematical identification because of the noise in the signal. NNs finds complex patterns from such inputs.
- the time is required only to train the model for NN. Once trained sufficiently, it's easy and fast to get an output for a given input fed to the network.

Motivation for using Neural Network

- . One of the biggest problems with Neural network is robustness and consistency. To address these problems (atleast partially), we will use Z3 in order to verify certain properties for which if the trained model is working correctly.

Data Generation

- To generate data sets, we have taken frequencies ranging from 1kHz to 5kHz

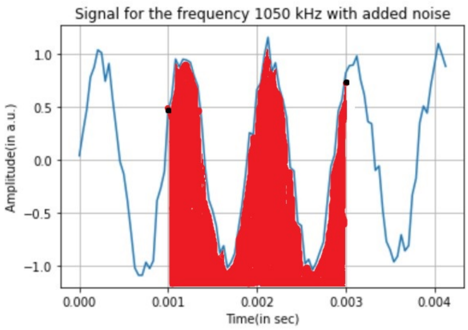
Data Generation

- To generate data sets, we have taken frequencies ranging from 1kHz to 5kHz
- The amplitude, A is set to 1. SNR is set to 25, as it is a typical setting for a good signal, thus $\sigma = 0.5$. The noise is calculated using Gaussian Distribution.

Data Generation

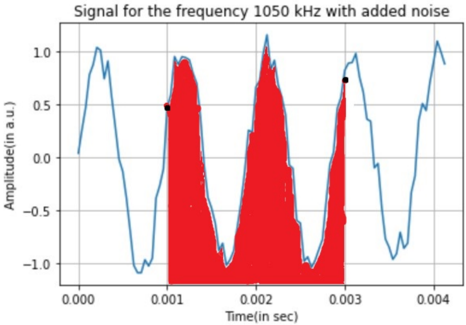
- To generate data sets, we have taken frequencies ranging from 1kHz to 5kHz
- The amplitude, A is set to 1. SNR is set to 25, as it is a typical setting for a good signal, thus $\sigma = 0.5$. The noise is calculated using Gaussian Distribution.
- The phase change is randomly assigned to the waves, in order to train our model for phase change.

Data Generation



- This is done by creating an array of 500 consecutive readings, with a time lapse of 4.1667 e^{-05}s, for a given signal, then randomly taking 50 consecutive readings from it and using that as our data. *Why taking like this?*

Data Generation



- This is done by creating an array of 500 consecutive readings, with a time lapse of 4.1667 e^{-05}s, for a given signal, then randomly taking 50 consecutive readings from it and using that as our data. *Why taking like this?*
- By this method we generated a dataset of size 2,00,000.

Data Generation

- To make the network free of any bias we introduce an extra input point that is initialized to 1 for any input.

Data Generation

- To make the network free of any bias we introduce an extra input point that is initialized to 1 for any input.
- This will simulate the shift that the biases intended.

Data Generation

- To make the network free of any bias we introduce an extra input point that is initialized to 1 for any input.
- This will simulate the shift that the biases intended.
- we normalize the data in order to have the output of the network range between 0 and 1.
- So one of the data in our dataset will look like: [[<The set of inputs = 50 points>,1],[<The normalized output>]]
  Why 50 points?

Data Separation

- From the 2,00,000 data we have taken, 70% as training data, 20% as validation data set, and 10% as the test data set, to train our Network.

Data Separation

- From the 2,00,000 data we have taken, 70% as training data, 20% as validation data set, and 10% as the test data set, to train our Network.
- We used Pandas to split the dataset that we generated randomly to get the training, testing and validation data.[using the sample and drop function.]

Data Formatting/Splitting

- We split each data from the dataset into its input and output counterparts.[Using delete and slice function]

Data Formatting/Splitting

- We split each data from the dataset into its input and output counterparts.[Using delete and slice function]
- Then convert the dataset which is a list to numpy array in order to flatten the data and reshape it in order to comply with the tensor format.

Data Formatting/Splitting

- We split each data from the dataset into its input and output counterparts.[Using delete and slice function]

Data Formatting/Splitting

- We split each data from the dataset into its input and output counterparts.[Using delete and slice function]
- Then convert the dataset which is a list to numpy array in order to flatten the data and reshape it in order to comply with the tensor format.
- Input Shape: (x, 51, 1) where x=140000(train), 20000(validation), 40000(test)
- Output Shape: (1,1)

The Network Configuration

- Our Neural Network have 3 layers.

The Network Configuration

- Our Neural Network have 3 layers.
- The first layer is the input layer, comprising 51 neuron. The three hidden layers have 9 neurons.

The Network Configuration

- Our Neural Network have 3 layers.
- The first layer is the input layer, comprising 51 neuron. The three hidden layers have 9 neurons.
- The last layer is the output layer with 1 neuron, that holds the predicted frequency(normalized).
- Here, We have used Relu as activation for all layers, where Relu can be defined as: $Relu(x) = Max(0, x)$

The Network Configuration

- Our Neural Network have 3 layers.
- The first layer is the input layer, comprising 51 neuron. The three hidden layers have 9 neurons.
- The last layer is the output layer with 1 neuron, that holds the predicted frequency(normalized).
- Here, We have used Relu as activation for all layers, where Relu can be defined as: $Relu(x) = Max(0, x)$
- The optimizer used is "Adam" optimizer.

Model Summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 51)                0
_________________________________________________________________
dense (Dense)                (None, 9)                 459
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 9
=================================================================
Total params: 468
Trainable params: 468
Non-trainable params: 0
_________________________________________________________________
```

Figure: Model Summary

Calculating Loss

- We have used MSE(Mean Squared Error) as a metric to measure the loss of the model.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_i - P_i)^2 \tag{1}$$

where n is the number of measurements, Y_i are the real values and P_i are the predicted values.

Loss graph

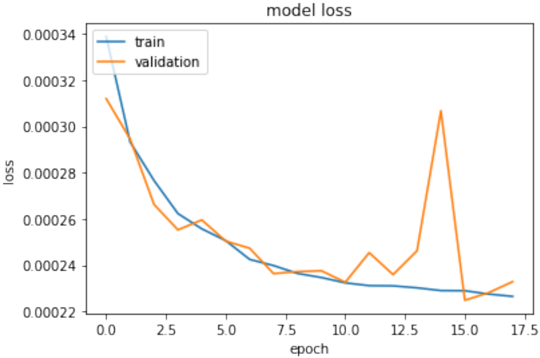


Figure: Loss Graph

Calculating Accuracy

- We have also measured its performance using accuracy with 5% as tolerance. The accuracy is calculated as follows:

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n}|P_i - Y_i| \leq T \times Y_i \tag{2}$$

  where n is the number of measurements, P_i is a predicted value, Y_i is the actual value and T is the threshold value. Here the threshold was set to 5%

- *Note: Here we are finding the average number of data points, for which our network predicts within the permissible threshold.*

Accuracy graph

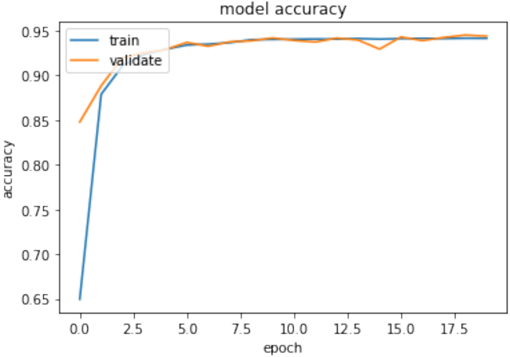


Figure: Accuracy Graph

Summary of the results

- We used 20 epochs for testing the model.
- loss: 2.2484e-04 - acc-net: 0.9420 - val-loss: 2.3291e-04 - val-acc-net: 0.9439

For Low Frequency

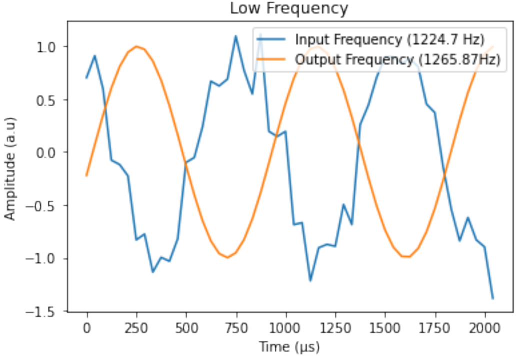


Figure: Behaviour of the model for Low frequency

For High Frequency

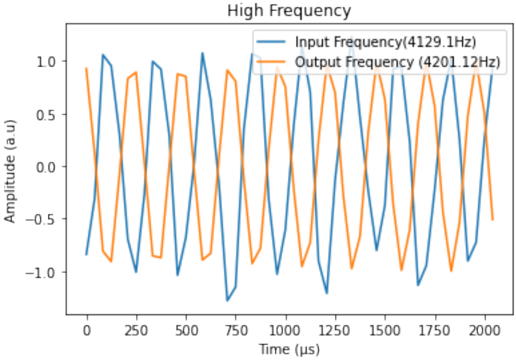


Figure: Behaviour of the model for High frequency

Z3 Verification

- To verify that the data given as an input is indeed a sinewave, we need to check whether the 50 input points form a probable sinewave.
- But we know that a sine function is neither built-in nor implementable in Z3.
- *So what should we do?*

Need to handle Sine function

What we can do is to approximate the sinewave by evaluating it using *Taylor Series Expansion* upto some terms so that we get a plausible signal.

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} \ldots$$

More problems...

- But even then we weren't able to get a good approximate when we tried to check the approximated function over a signal $S = A\sin(2\pi ft + \phi)$.
- Then, we realized that the value that sin would evaluate can be $> 2\pi$ so in order to generalize it to some value that is within the range 0 to 2π of a sine function we implemented a MOD simulation in Z3 (since Z3 doesn't have a MOD function) so that we are able to get a meaningful result.
- We took the approximation to be upto 8 terms and got a good estimate.

Deriving the Z3 Model

- In order to get the predicted value of the Neural Network, We need to model the Trained Network using Z3.

Deriving the Z3 Model

- In order to get the predicted value of the Neural Network, We need to model the Trained Network using Z3.
- To achieve that, we extracted the weights from the model that we trained and used a method GET-MODEL() which invokes a feedforward function complying with Z3 variables, such that when given the weights the method will output a solver,an optimizer, an input and an output, where the input and output are basically Z3 variables and the solver would hold the constraints that our Neural Network learned while training and the optimizer will also hold the same constraints.

Deriving the Z3 Model

- *Then why use both optimizer and solver?*
- We want to verify some maximizing properties like the maximum phase change so that the frequency remains in a given threshold, and optimizer allows us to use a built-in maximize function that can take a z3 variable and maximize its values so that all the constraints still holds.

Property 1: Phase Change

- The phase of a signal S is the angle ψ, in degrees or radians that the waveform has shifted from a certain reference point along the horizontal zero axis.
  We define the verification problem as:
  What is the maximum change allowed in the phase of the signal so that the Network still predicts the frequency within a given threshold?

Property 1: Phase Change

- The threshold that we took was $\pm 5\%$ of the original frequency.
- In order to do the verification(s), we took the weights from the Trained Network and fed it to the Z3 function $\textsc{get\_model}()$ as described.
- This way, we get the input variables and output variable, a solver and an optimizer that constraints the input and output to comply with our trained Network.

Property 1: Phase Change

- The input is then constrained with what was described (i.e., a function that approximates a sinewave).
- We also put a constraint that the range of the sampling points(i.e., the input variables) must be between -1 to +1 as the amplitude is fixed to 1.
- We constrain the frequency to be between the permissible range of 1KHz to 5KHz.
- We took the same variable with the same constraint over the optimizer and added the maximize constraint on the phase variable and checked the model.

Property 1: Phase Change

- The input is then constrained with what was described (i.e., a function that approximates a sinewave).
- We also put a constraint that the range of the sampling points(i.e., the input variables) must be between -1 to +1 as the amplitude is fixed to 1.
- We constrain the frequency to be between the permissible range of 1KHz to 5KHz.
- We took the same variable with the same constraint over the optimizer and added the maximize constraint on the phase variable and checked the model.
- : (

Property 1: Phase Change

- So we tried another variant of the same problem where the verification is **whether there is a phase value that results in the frequency to go beyond the threshold where the frequency is not fixed.**
- Now in order to address the later problem, we have taken the phase to be a Z3 variable and constrained it to be MOD value with respect to 2π and checked the model where every constraint was taken over the solver.

Property 1: Phase Change

- So we tried another variant of the same problem where the verification is **whether there is a phase value that results in the frequency to go beyond the threshold where the frequency is not fixed.**
- Now in order to address the later problem, we have taken the phase to be a Z3 variable and constrained it to be MOD value with respect to 2π and checked the model where every constraint was taken over the solver.
- : (

Property 1: Phase Change

- So we tried another variant of the same problem where the verification is **whether given a frequency value, is there a phase difference that results in the frequency to go beyond the threshold.**

Property 1: Phase Change

- So we tried another variant of the same problem where the verification is **whether given a frequency value, is there a phase difference that results in the frequency to go beyond the threshold.**
- :)

For Phase Change

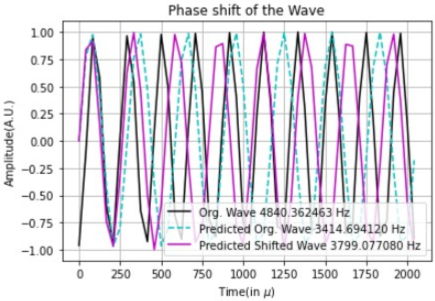


Figure: Phase Change

Property 2: Reverse

- Property: Here we want to check if frequencies are independent of the direction in which, signal is read.
- That is, if we reflect the signal then the frequency predicted by our model remains the same or not.
- To check this property, the sample points are reversed and fed as input to our model. We first generate signal using z3 approximated sine wave.
- Let the generated signal be of the form $[a_0, a_1, ...a_{50}]$, then $[a_{50}, a_{49}, ..., a_1]$ is fed as input to our Z3network.

Property 2: Reverse

- Similar to above verification, we ensure our variables comply with our trained model by using Z3 function GET_MODEL(), also , frequency and input is constrained within the range .
- Here we used solver to find if there exists an input for which the predicted output is not within the threshold.

Property 2: Reverse

- Similar to above verification, we ensure our variables comply with our trained model by using Z3 function $\textsc{get\_model}()$, also , frequency and input is constrained within the range .
- Here we used solver to find if there exists an input for which the predicted output is not within the threshold.
- : (

Property 2: Reverse

- So we tried another variant of the same problem where the verification is **whether given a frequency value, Does the reverse of the wave results in the frequency to go beyond the threshold.**

Property 2: Reverse

- So we tried another variant of the same problem where the verification is **whether given a frequency value, Does the reverse of the wave results in the frequency to go beyond the threshold.**
- :)

For Reversal

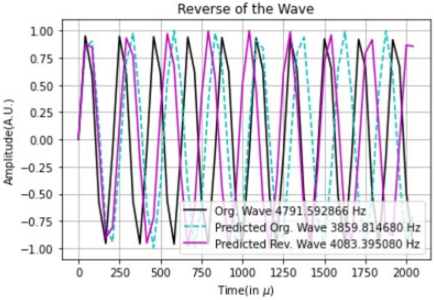


Figure: Reversal of the Wave

Property 3: Flipping

- We define 'flipping' of a wave, as its reflection with respect to x-axis.
- Property:Here we want to check if there exists a frequency,f, for which when the signalis flipped and given as input to our network , the network outout is greater that the tolerance.
- To check this property, the sign of the sample points are flipped, and fed as input to our model.
- We first generate signal using z3 approximated sine wave. Let the generated signal be of the form $[a_0, a_1, ...a_{50}]$,then $[-a_0, -a_1, ..., -a_{50}]$ is fed as input to our Z3network.
- Constraints over neural network is similar to the above verification. Frequency is constrained within the range.
- Here we used solver to find if there exists a frequency for which the predicted out is not within the threshold.

Property 3: Flipping

- We define 'flipping' of a wave, as its reflection with respect to x-axis.
- Property:Here we want to check if there exists a frequency,f, for which when the signalis flipped and given as input to our network , the network outout is greater that the tolerance.
- To check this property, the sign of the sample points are flipped, and fed as input to our model.
- We first generate signal using z3 approximated sine wave. Let the generated signal be of the form $[a_0, a_1, ...a_{50}]$,then $[-a_0, -a_1, ..., -a_{50}]$ is fed as input to our Z3network.
- Constraints over neural network is similar to the above verification. Frequency is constrained within the range.
- Here we used solver to find if there exists a frequency for which the predicted out is not within the threshold.
- :)

For Flipping

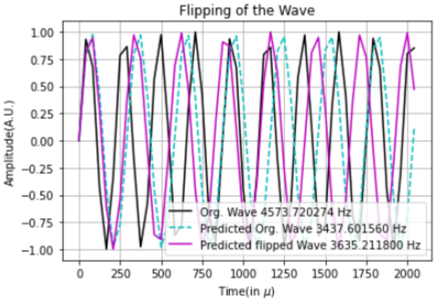


Figure: Flipping of the Wave

Property 4: Amplitude

- For a given signal the amplitude of a signal is the maximum displacement of the signal with respect to the x-axis.
- Property: Any change in the amplitude doesn't affect the frequency of the signal.
- We try to find what is the minimum change in the amplitude that leads to the a change in the frequency by some threshold(=5% of the upper range of the frequency that the network can verify), if any.
- Since it uses optimize function and the calculation is quite heavy for Z3. We try to check a lesser constrained variant of the above problem where we check whether there exists some amplitude other than the given one for which the change in the frequency exceeds the threshold limit using the solver.

For Change in Amplitude

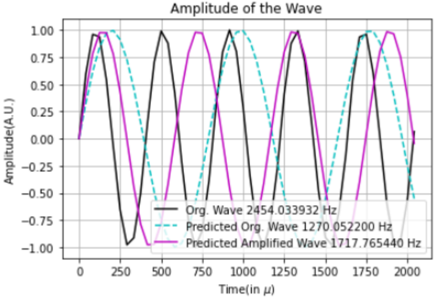


Figure: Changing Amplitude of the Wave

CEGAR Implementation

...