Complex Systems Informatics and Modeling Quarterly (CSIMQ)
eISSN: 2255-9922
Published online by RTU Press, https://csimq-journals.rtu.lv
Article 135, Issue 23, June/July 2020, Pages 44–57
https://doi.org/10.7250/csimq.2020-23.04

Sinusoidal Neural Networks: Towards ANN that Learns Faster

Tekin Evrim Ozmermer*

Department of Artificial Intelligence and Systems Engineering, Riga Technical University, 1 Kalku Street, Riga, LV-1658, Latvia

evrimozmermer@hotmail.com

Abstract. If everything is a signal and combination of signals, everything can be represented with Fourier representations. Then, is it possible to represent a signal with a conditional dependency to input data? This research is devoted to the development of Sinusoidal Neural Networks (SNNs). The motivation to develop SNNs is to design an artificial neural network (ANN) algorithm that can learn faster. A short review of the history of biological neurons helps to identify components that should be redesigned in ANNs. After the components are identified, a new neural network algorithm called SNN is proposed. Experiments are conducted to show the practical results of the algorithm. According to the experiments, the proposed neural network can reach high accuracy rates faster than the standard neural networks, while an interesting generalization capacity is obtained for the developed algorithm. Even though the promising results are achieved, further research is necessary to test if SNNs are capable of learning faster than existing algorithms in real-life cases.
Keywords: Artificial Neural Networks, Fourier Neural Networks, Periodic Functions, Activation Function, Node Operation.

1 Introduction

Deep neural network (DNN) algorithms are being used as solutions to problems from various research and business areas. Although DNN algorithms let the research community handle many tasks, they are dependent on data, even more than other machine learning algorithms. Most of the solutions which use DNNs are using models that learn with supervision. Supervised learning is a learning method where the neural networks learn to map given input data to given output data, which requires each sample in the training set to be labeled by humans. In the solutions which require supervised learning DNNs should have not only a high amount of data, but also labeled data. This situation created a new market where the data is valuable and computation resources that can train the DNNs faster are on demand.

Speed of training and inference operations for DNNs are crucial. It is almost a must to use a GPU computation unit for DNN models in the production level. However, these GPU

* Corresponding author

Reference: T. E. Ozmermer, "Sinusoidal Neural Networks: Towards ANN that Learns Faster," Complex Systems Informatics and Modeling Quarterly, CSIMQ, no. 23, pp. 44–57, 2020. Available: https://doi.org/10.7250/csimq.2020-23.04

Additional information. PII S225599222000135X. Received: 31 May 2019. Accepted: 23 July 2020. Available online: 31 July 2020.

computation units are more expensive than the regular CPU computation units which are used as web servers. In March 2016, a reinforcement learning model so-called AlphaGo has won against Lee Sedol in the game of Go. AlphaGo has created a big impact and awareness to the neural networks. On the other hand, it was also emphasized that DNNs can only be the tool of businesses rather than individuals. An analysis [1] calculated the cost of AlphaZero, a new version of AlphaGo. To train the AlphaZero neural network model to reach the same level with AlphaGo that has won against Lee Sedol, three days and approximately 35 million dollars are necessary only for the payment of computation units [1]. This example not only shows the lack of opportunity for using DNNs by individuals but also highlights the necessity of the methods and algorithms which can reduce the learning time thus reducing the expenditures made on computation power. Therefore, improvements which are going to speed up the learning process and will make the neural network algorithms require fewer data will help to researches and businesses.

Improvements for neural network algorithms can be done on several components of neural networks such as activation functions, learning rate calculations, backpropagation calculations, and network architectures. For improvements, node operation of neural networks is not changed frequently. Therefore, linear equation (1) is used as the node operation of neural networks [2].

$$\sigma(x) = \sum_{n=1}^{k} w_n x_n + b \tag{1}$$

Where

x is the input of neurons,

n is a neuron indicator,

w is the weight (multiplier) of a neuron,

b is the bias.

In this article, the node operation is intended to be changed to improve the neural network algorithm. To define the new node operation, a review of biological neurons, and an assumption about the behavior of signal transmission in biological neurons are discussed in Section 2. After that, the assumed behavior of signal transmission in biological neurons is used to define new node operation in Section 3. This way, we are confident to take an important step towards an artificial neural network (ANN) that learn faster. Experiments are presented in Section 4. Section 5 provides brief conclusions.

2 Background

2.1 Biological Neurons

Biological neurons are the building blocks of the information processing units of all species. Although the cellular structure of the biological neurons is known, the working principle of neurons enabling the learning process is still unclear. In this section, some concepts of biological neurons are reviewed. After that, the behavior of the action potential concept is put in relation to activation and node operation components of artificial neural networks.

Axon

Axon is the long neuronal process that ensures the conduction of information from the cell body to the nerve terminal. German anatomist Otto Friedrich Karl Deiters first distinguished the axon from the dendrites. The axon initial segment was originally identified by the Swiss Rüdolf

Albert von Kölliker and the German Robert Remak [3]. It is the extension of a neuron to another neuron's dendrites transmitting the information. Axons can be with myelin and without myelin. There can only be one axon in one neuron.

Myelin Sheath

"Myelin sheath is a layer that is made of proteins and fats. Myelin sheaths insulate the axons of neurons from outside of the neural cells [4]. Myelin sheath speeds up the action potential conduction in an axon. Therefore, the information is transmitted faster from the presynaptic neuron to the postsynaptic neuron.

When myelin sheath covers the axon (see Figure 1), "transmembrane currents can only occur at the nodes of Ranvier where the axonal membrane is exposed". Therefore, the action potential occurs only in Ranvier nodes. [5]. It can be imagined as the myelin sheath is a jumping ramp for the information that flows through the axon.

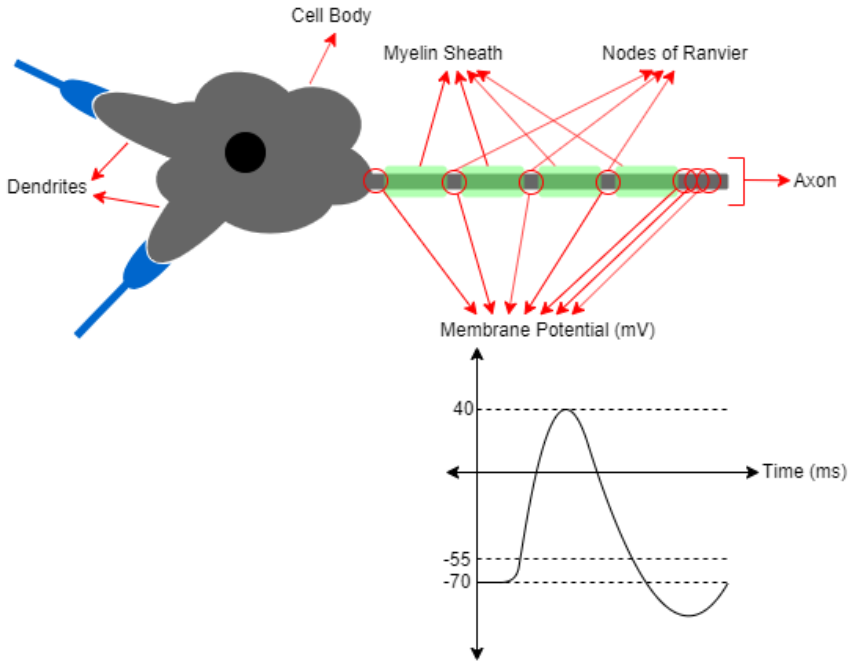

Figure 1. A basic visualization of a biological neuron and graph of membrane potential

Action Potential

Action potentials are activities which are triggered by the incoming signals coming from dendrites and cell body, respectively. The signals that are received from dendrites transmit through the membrane of the cell body. These signals reach the axon and trigger the special types of voltage-gated ion channels embedded in a cell's plasma membrane. When the membrane potential is under a defined threshold, the voltage-gated sodium channels are closed. When the membrane potential is above the defined threshold, voltage-gated sodium channels are open. When sodium channels are open, sodium ions, which are located outside of the cell, get into the axon. This triggers a chain reaction where the increasing potential triggered by sodium channels triggers the nearby sodium channels. When the membrane potential reaches to 40 mV, voltage-gated sodium channels close. After voltage-gated sodium channels are opened, voltage-gated potassium channels which take the potassium ions (K^+) out of the cell also open with a delay. When potassium channels are opened, the sodium channels are closed. Therefore, membrane potential starts decreasing. Potassium channels start closing when the membrane potential descends. This activity takes some time. Therefore, membrane potential gets lower than resting potential. During the opening and closing activities of voltage-gated sodium and potassium channels, sodium-potassium pumps continuously take potassium ions in and evacuate

sodium ions out. This process stabilizes the amount of sodium and potassium ions in the cell and, also fixes the potential overshooting of the voltage-gated potassium channel [6], [7], [8].

Sinusoidal Signals in Information Transmission

In conclusion, dendrites of the cell take the signals in the cell body. Signal flows through the membrane of the cell and reaches the axon. If the signal is powerful enough, the action potential starts. Signal flows through axon and Ranvier Nodes (if there is myelin sheath around axon) and the signal reaches the synapses which connect to dendrites of the postsynaptic neuron.

The most important fact in information flow is action potential. If a sinusoidal signal is bounded in a specific region, a signal similar to action potential can be obtained. When a signal in axon reaches to the synapses, the voltage value can be any point on the signal of the action potential. In this research, the author assumes that the phenomena of action potential have the behavior of a sinusoidal signal. Length and width of the axon, number of Ranvier nodes, length of telodendria (end of an axon) determines the voltage value that is transmitted to the synapses. As a result, it is assumed that the transmission of information between neurons is made via sinusoidal signals which can be adjusted by the features and components of the axon.

Because the transmitted signals are sinusoidal signals, the neural networks which are using periodic functions such as sine and cosine as node operation or activation function are pursued to be developed.

2.2 Sine as Activation Function and Node Operation

The non-linear functions which are commonly used in neural networks are tanh, sigmoid, and softmax which are monotonic functions. “It is well known that units with monotonic activation functions generate a single hyperplane which divides the input space into two regions” [9]. This brings a case that the neural network is forced to approach input space to a bounded region. This case ends up with neural networks that first approach the bounded region before they start mapping the inputs to the expected output, therefore the training process becomes slower. If there was an activation function that prevents this artifact, that function could help neural networks train faster. The candidate activation function is found to be a periodic function, because periodic functions can divide the input space to an infinite number of regions.

The use of periodic functions in neural networks is not new. In 1999, Josep M. Sopena published a paper [9] that proposes a neural network algorithm that uses sine function as the activation function in neural networks. The paper [10] investigates the use of sine as the activation function and proves the ability of sine function as the activation function. The confusing fact for the researches on this subject is naming the purpose of sine. The way that sine is executed determines its purpose. While it can be used as an activation function, it can also be assumed that it is used as the node operation. In this research, it is accepted that if the sine is executed on top of sum of node operations, it is used as the activation function, while the sine that is executed just after every node operation $w \times x + b$ (before summation of neuron outputs) is used as the node operation.

The sine activation function lets the neural network to have alternative input values for the same output value. Therefore, if it is thought from a probability perspective, the probability value 1 can be obtained with an infinite number of input values (Figure 2).

As can be seen from Figure 2, sine function can get any output value with an infinite number of alternative input values because it is a periodic function, whereas the sigmoid function can divide the space only to two regions.

[9] researches the use of sine function as the activation function. The paper stresses that the infinite number of divided spaces can help neural networks to train faster. Especially the experiment made with two spiral dataset shows the strong impact of the sine activation function. The authors claims that “with the correct range (range refers to the range of initial weights), the learning speed obtained with sine functions is 40 times greater than the fastest current method” [9]. In the conclusion section of the paper, the authors state that the similarities between neural

network models with sine activation function and Fourier series bring an opportunity to interpret weights in terms of frequencies. The sine activation function shows that it can be used not only as an activation function that divides the space into infinite regions but also to find a sum of sine functions that create Fourier series-like representation of input-to-output mapping.

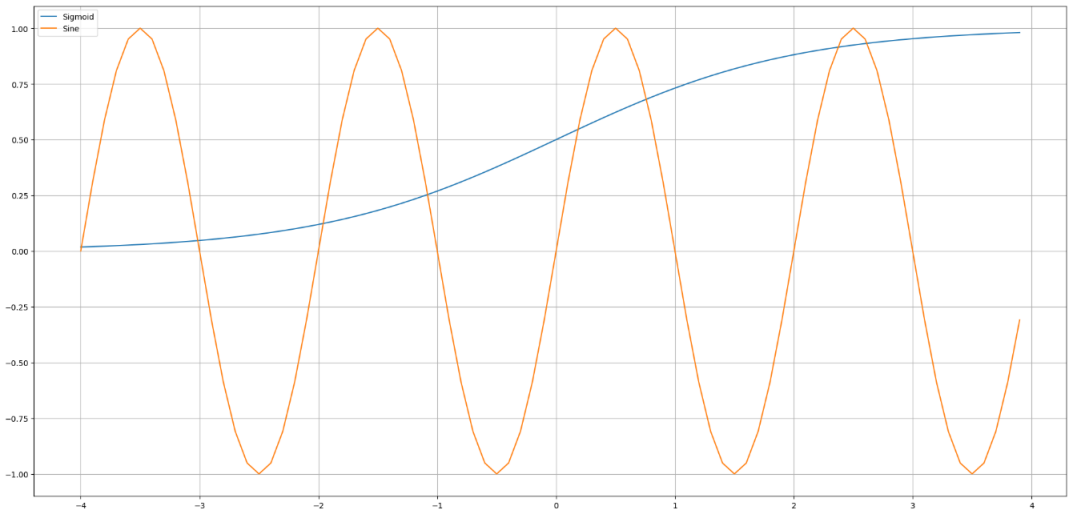

Figure 2. Output of artificial neural networks activated with sine function and sigmoid function

In the paper [10], the experiments made with the MNIST [20] dataset has shown that the use of sine function and tanh function as the activation has not given different accuracy rates. This may be due to the fact that the tanh function and sine function behave similarly for small values.

Sine has been used not only as an activation function but also as a node operation. The neural network algorithms using sine as node operation were called Fourier Neural Networks (FNNs) in previous researches, namely [19], [11], [14]. In the paper [15], the performance of three FNNs is compared. The paper shows and states that none of the FNN algorithms outperforms the vanilla neural network model (standard neural network which has linear node operation that is activated with sigmoid function). The FNN algorithms which are examined in the [15] are FNN of Gallant and White [19], FNN of Silvescu [11], and FNN of Liu [14].

Gallant and White published their paper "There exists a neural network that does not make avoidable mistakes" in 1988 [19]. This was the earliest research related to the subject of Fourier-like neural networks. In the paper, the authors suggested a modified version of cosine function as the activation function in the artificial neural network. They used the term "Cosine Squasher" for the formula (4).

$$\begin{cases} 0 & x \in \left(-\infty, -\frac{\pi}{2}\right), \\ \frac{1}{2}\left(\cos\left(x + \frac{3\pi}{2}\right) + 1\right), & x \in \left[-\frac{\pi}{2}, +\frac{\pi}{2}\right], \\ 1 & x \in \left(+\infty, +\frac{\pi}{2}\right), \end{cases} \tag{4}$$

Adrian Silvescu published his paper "Fourier Neural Networks" in 1999 [11]. The author proposed a different algorithm than conventional ones. The node operations held in each node in the layers were not summed as it was in previous algorithms. In the proposed algorithm, the multiplication of cosines took the place of the sum of linear functions. The author uses cosine representation for $e^{i(y_1x_1+\cdots+y_nx_n)}$. After that, the sigmoid function is applied to get values between 0 and 1 as the output for the function. Hereby, the function (2) is derived [11].

$$f(x_1, \dots, x_n) = \text{sigmoid}\left(\sum_i c_i \prod_{j=1}^{n} \cos\left(w_{ij} x_j + \varphi\right)\right) \quad (2)$$

Another type of FNN is proposed in the paper [12]. The function of node operation is more complex than the previous ones. It uses sine and cosine functions together like it is a Fourier series representation. The function can be seen in formula (3.1) The interesting fact in this function is that the frequency parameters of sine and cosine are the same. This situation will restrict the function to go to zero only when the amplitude of sine and cosines goes to zero. This may solve the vanishing gradient problem if the amplitudes of sine and cosine are given big enough. On the other hand, it may cause a situation where the function cannot converge to lower error rates. It is also necessary to indicate that sine and cosine functions are equivalent to each other with shifted phase $\pi/2$. Therefore, the function can also be written as (3.2) or (3.3). In this case, the same term is duplicated with a shifted phase and multiplied with different factors. The function is taken from [13].

$$x \to v_0 + \sum_{k=1}^{n} v_k \cos(w_k x + b_k) + u_k sin(w_k + b_k) \quad (3.1)$$

$$x \to v_0 + \sum_{k=1}^{n} v_k \sin(w_k x + b_k - \pi/2) + u_k sin(w_k + b_k) \quad (3.2)$$

$$x \to v_0 + \sum_{k=1}^{n} v_k \cos(w_k x + b_k) + u_k cos(w_k + b_k + \pi/2) \quad (3.3)$$

The most complex and effective version of the proposed FNNs is the FNN of Liu [14]. The function is similar to the FNN of Tan, Zuo, Cai. The only difference is that the frequency parameters of sine and cosine are not the same. Therefore, including the amplitudes, there are six parameters to adjust. Formula (4) is taken from [15].

$$x \to v_0 + \sum_{k=1}^{n} v_k \cos(w_k x + b_k) + u_k sin(p_k + q_k) \quad (4)$$

Two spirals experiment that is conducted in [14] shows that this FNN has the remarkable capability to represent this complex dataset perfectly. The results are considered good. The receptive field that is created by FNN of Liu for two spirals problem is shown in Figure 6. This shows that the FNNs have a potential for developing deep neural network models with capabilities of high accuracy and fast performance.

There are successful and failed examples of FNNs. When sine function is used as an activation function; the infinite space division is aimed. On the other hand, when sine function (or cosine function or both together) are used as node operation in FNNs, Fourier representation of input-to-output mapping is aimed. In this article, the sine is used as node operation where it is analyzed and redesigned using unit circles. Because the functional behavior of a neural network using sine function as node operation can be visualized using unit circles. In the remainder of the article, a neural network using sine function is called a sinusoidal neural network (SNN).

3 Designing the Node Operation

Through the research, it is decided to visualize the functional behavior of the sine function in neural networks by using the unit circle, as it helps to understand the behavior of sine function easier [16]. The unit circle is a circle where the center of the circle having value 1 as radius sits

on the origin of a coordinate plane. Each quarter in the circle has angle π/2 and, the x-axis of the circle represents cosine while the y-axis of the circle represents sine. All of the trigonometric functions of the angle θ (theta) can be constructed geometrically in terms of a unit circle centered at O [17] (see Figure 4).

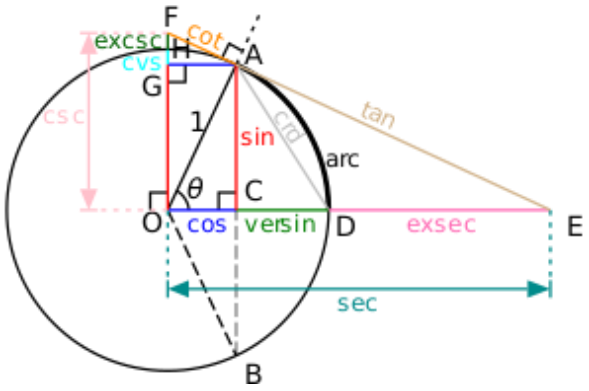

Figure 3. Unit circle demonstrating trigonometric functions [18]

By using the unit circle, the behavior of the sum of sine functions which create the general mapping function of the sinusoidal neural networks can be seen in Figure 5 for formula (S.1), in Figure 6 for formula (S.2) and, Figure 7 for formula (S.3).

$$y_{s1} = \sin(x\ \pi\ w_i) \quad \text{S.1}$$

$$y_{s2} = \mathrm{w_o} \sin(x\ \pi\ w_i) \quad \text{S.2}$$

$$\mathrm{y_{s3}} = \mathrm{w_o}\ x \sin(x\ \pi\ w_i) \quad \text{S.3}$$

The derivative of the functions is used for backpropagation. Therefore, it is also necessary to mention them. The derivatives of S.1, S.2, and S.3 with respect to their parameters can be found in S.b.1, S.b.2, S.b.3, respectively. The function labels of derivatives are numbered with n as S.b.1.n. S.b.1.1 represents the derivative of S.1 with respect to w_i, S.b.1.2 represents the derivative of S.1 with respect to x, S.b.2.1 represents the derivative of S.2 with respect to w_i, S.b.2.2 represents the derivative of S.2 with respect to w_o, S.b.2.3 represents the derivative of S.2 with respect to x, S.b.3.1 represents the derivative of S.3 with respect to w_i, S.b.3.2 represents the derivative of S.3 with respect to w_o, S.b.3.3 represents the derivative of S.3 with respect to x.

$$\frac{dy_{s1}}{dw_i} = \pi\ x \cos(x\ \pi\ w_i) \quad \text{S.b.1.1}$$

$$\frac{dy_{s1}}{dx} = \pi\ w_i \cos(x\ \pi\ w_i) \quad \text{S.b.1.2}$$

$$\frac{dy_{s2}}{dw_i} = \pi\ x\ w_o \cos(x\ \pi\ w_i) \quad \text{S.b.2.1}$$

$$\frac{dy_{s2}}{dw_o} = \sin(x\ \pi\ w_i) \quad \text{S.b.2.2}$$

$$\frac{dy_{s2}}{dx} = \pi\ w_i\ w_o \cos(x\ \pi\ w_i) \quad \text{S.b.2.3}$$

$$\frac{dy_{s3}}{dw_i} = \pi\ x\ w_o \cos(x\ \pi\ w_i) \quad \text{S.b.3.1}$$

$$\frac{dy_{s3}}{dw_o} = \pi\ x\ w_o \sin(x\ \pi\ w_i) \quad \text{S.b.3.2}$$

$$\frac{dy_{s3}}{dx} = w_o \sin(x\ \pi\ w_i) + \pi\ x\ w_i\ w_o \cos(x\ \pi\ w_i) \quad \text{S.b.3.3}$$

In formula S.1, there is only one weight (w_i) which controls the phase (angle) of the sine function. In the unit circle, it is the rotation amount of the vector. As can be seen from the Figure 5 of formula S.1, the amplitude of the sine is fixed to value 1. Because the adjustments can be done only on the rotation-wise, it is harder to approximate a function with this formula. If the samples in the dataset that we use to train the neural network are not spread to a wide range, this formula can approximate the output function. Otherwise, if samples are distributed to a wide range, the formula is not capable to approximate the output function with high accuracy.

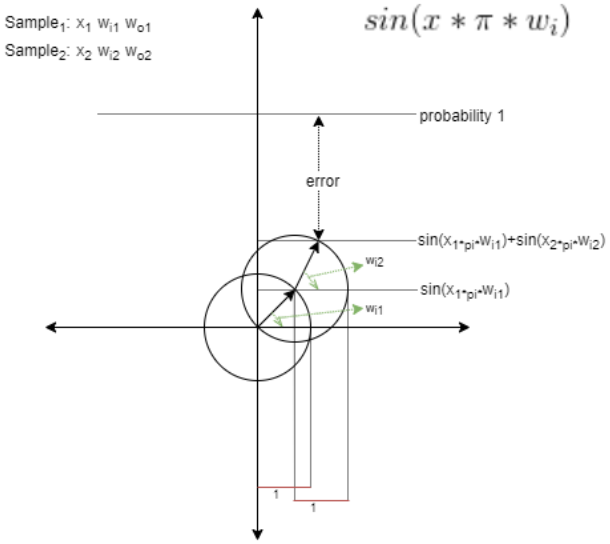

Figure 4. Representation of S.1 with unit circles that are combined end-to-end

In formula S.2, two weights adjust the phase and amplitude of the sine function. If it is represented with the unit circle, it can be seen that the amplitude of the sine function is not limited to value 1 as in formula 1. Thus, the adjustments, which will be done to lower the error, can be made on both weights. This lets the neural network approximate more complex functions that map dataset with samples distributed to a wide range of values. Besides, it is necessary to indicate that the initial weights for w_o can also be negative. From the unit circle representation, it may seem like the negative values for w_o will cause a negative radius for each unit circle. But this is not true. When the values for w_o are negative, unit circle take an upside-down position, which refers to a 2π phase shift.

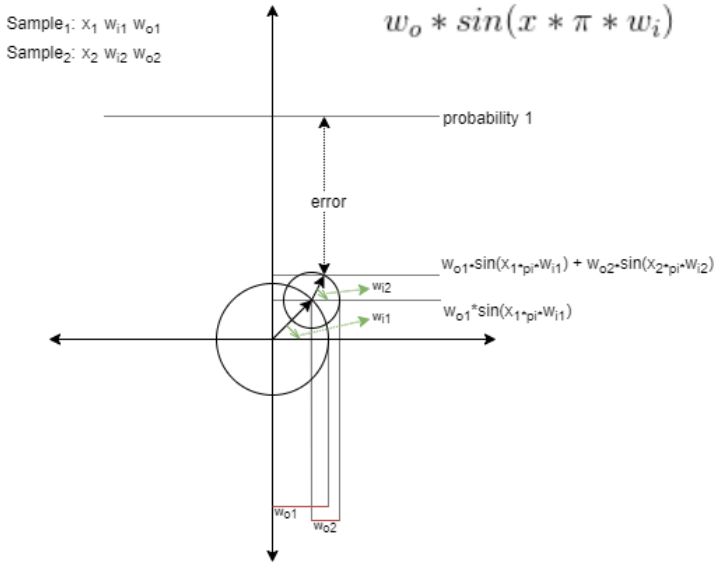

Figure 5. Representation of S.2 with unit circles that are combined end-to-end

Formula S.3 is slightly modified formula S.2. The only difference of it is that the input value x also affects the amplitude of the signal. Theoretically, this increases the ability to approximate more complex functions. The potential drawback is that this may also raise the uncertainty of the output function.

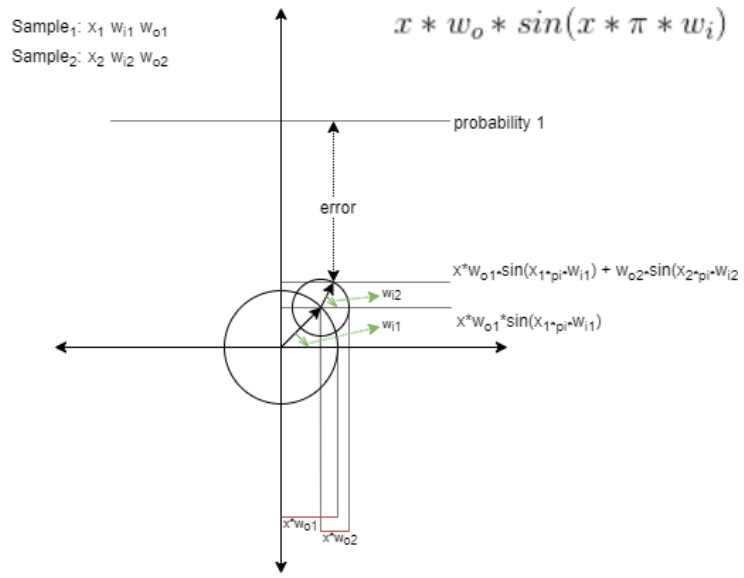

Figure 6. Representation of S.3 with unit circles that are combined end-to-end

To see the practical results of the SNNs, experiments with two datasets are conducted. In addition to the SNNs, a common artificial neural network algorithm using the linear function as node operation and one of sigmoid, softmax, or tanh functions as activation are used to compare the results. The common artificial neural network algorithm is called a standard neural network in experiments.

4 Experiments

Experiments are conducted to compare SNNs with three formulas that are mentioned above (S.1, S.2, S.3) and a standard neural network. Initial weights of SNN are distributed with normal distribution, and they are normalized to a range [-0.1,0.1] while -0.1 is the minimum value and 0.1 is the maximum value for normalization. SNN does not have any activation function in the hidden and output layers. The error is calculated by subtracting the target value from the output of the neural network.

The standard neural networks differ for each dataset. For MNIST [20] dataset, the hidden layer is activated with ReLU [18] and the output layer is activated with softmax function while the error is calculated with a categorical cross-entropy function. For two spirals, the hidden layer is activated with ReLU, and the output layer is activated with tanh function while the error is calculated with mean squared error function. These parameters are chosen because they let the model reach to the highest accuracy.

For both SNNs and standard neural network, the stochastic gradient descent method is used to backpropagate the calculated errors. The batch size for training is 1.

For experiments, Python language is used to develop. To build a standard neural network, Tensorflow/Keras library is used. Tensor operations from the PyTorch library are used to build the SNN from scratch. The code of experiment for SNN and standard neural network can be found in the GitHub repository in the link [21].

4.1 MNIST

In the first experiment, MNIST [20] dataset is used. The MNIST dataset is the dataset composed of hand-written images, and it is mostly used to test the effectiveness of a new algorithm in machine learning. It has 60 000 images for training and 10 000 images for testing. A feed-forward neural network model having one hidden layer with 128 nodes (neurons) is used to test the newly designed node operation formulas. There are no activation functions applied to the neural network and the error rate is calculated by only subtracting the expected output from the output of the neural network. The reason for not applying appropriate error functions is to restrict the neural network from any other optimizers. For the backpropagation, a stochastic gradient descent algorithm is used. The accuracy rate is the metric that is used to assess the success of the formula. The results of the experiment can be seen in Table 1.

Table 1. Test results from the MNIST experiment

Formula	Accuracy (1 Epochs)	Accuracy (5 Epochs)	Accuracy (10 Epochs)
S.1	90.8%	92.7%	93.4%
S.2	90.7%	92.8%	93.8%
S.3	93.6%	96.16%	96.9%
Standard	93.3%	97.5%	97.6%

As it can be seen from the test results, the third algorithm has reached to 93% accuracy rate much faster than the first and the second algorithms, because the third algorithm is using more parameters that can be adjusted for mapping the input to output. It was expected that the S.2 would give considerably better results than the first function because the multiplier of sine enables the neural network to adjust the amplitude of sine functions as well. Adjusting the amplitude of sine could enable the model converge to a smaller error rate with a fewer number of nodes. On the other hand, it can be seen that the standard neural network has similar results with SNN S.3. On top of that, it reaches to accuracy rates higher than all SNNs for further epochs. It is necessary to mention that not any special error function or activation function are used in neural networks with sine node operation. With an appropriate error function, the neural networks with sine node operations could reach higher accuracy rates with the same number of epochs.

4.2 Two Spirals

Another experiment is conducted with two spiral problem. Two spiral problem is a hard problem to solve for neural networks. Therefore, since the beginning of neural network researches, two spiral problem is used as a benchmark to evaluate neural network algorithms. In the problem, two spirals are intertwined, and the dataset of the problem includes x and y coordinates of the points in each spiral as input and 0 and 1 as labels for each spiral as output. The neural networks with sine node operations employing each formula and a standard neural network are used in the experiment. Two metrics are used to evaluate the models. The first metric is the accuracy rate, the second metric is the receptive field. The reason for having a receptive field as an evaluation metric is to see the generalization capability of models to classify values which are not included in the dataset. All neural network models are using 2-16-16-1 architecture, where 2 is the number of input nodes, 16 is the number of nodes in hidden layers and 1 is the number of output nodes. The neural networks are trained with a mean squared error. For training and testing purposes, two distinct datasets having 2000 samples and including normally distributed random values in the range [0,0.05] as noise are created. Noise values that are added to the training and testing dataset are different. To create a receptive field, a two-dimensional matrix is created and row and column values in the two-dimensional matrix are normalized to the range [0,1]. To observe the neural network output that is tested with values that are outside of the range [0,1], another

receptive field in the range [-1.5,1.5] is created by shifting and scaling the receptive field in the range [0,1]. The accuracy rates can be seen in Table 2.

Table 2: Test results from two spiral experiment

Formula	Accuracy (Epoch: 50)
S.1	52.3%
S.2	97.1%
S.3	92.9%
Standard	71.1%

As it can be seen from Table 2, S.2 achieved the best accuracy rates while the S.1 could only reach 54.1% accuracy. S.3 has achieved a high accuracy rate, but not as good as the S.2. On the other hand, the standard neural network could reach only 59.3% accuracy. It was expected that the formula S.3 achieves the best accuracy. Each sample that is put into the calculation in SNN with S.3 disrupts the amplitude of sine functions in the neural network. Therefore, the model is supposed to be more robust to various inputs and generalize better after it has been trained. By plotting the receptive field, the behavior of neural networks is going to be easier to be understood.

Figure 8 demonstrates the receptive field of the neural network using S.1. The model can create non-linear distributions, but the distributions are wrong. The reason behind it is that the first formula does not have the multiplier in front of sine. Therefore, each sine function in the model has a fixed radius from the unit circle perspective. A fixed radius restricts the model to adjust the amplitude of sine to adapt to necessary conditions.

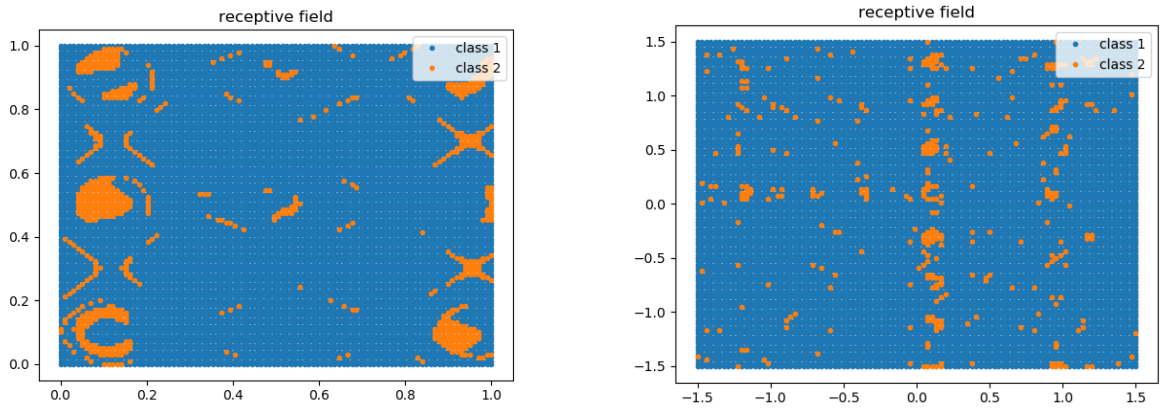

Figure 7. Receptive field of sinusoidal neural network using S.1

Figure 9 shows the receptive field of the neural network using S.2. The receptive field of the model of this formula gives understandable outputs. It clearly divided the space as it was trained. Even though it has the best accuracy rates in the test results, its receptive field shows that its generalization capacity for values that are out of the range of training samples is low, because the values which are not included in samples are almost randomly divided. The deeper investigation of these functions is a further research topic for this subject.

Figure 10 demonstrates the receptive field of neural network S.3. This model has not achieved the best results by means of accuracy. But its generalization capacity for values that are out of the range of training samples is better than for the model with the S.2. As can be seen from Figure 10, the model divided the field clearer. On the other hand, the enlarged and shifted receptive field on the right side of Figure 10 shows that the neural network has learned to repeat the pattern it learned for minus values as well. The model perfectly reflects the division that it learned also for minus values.

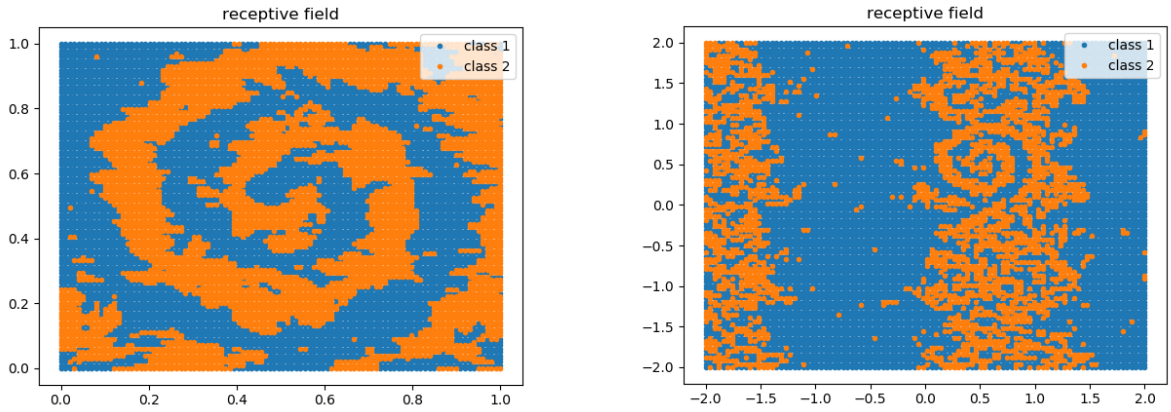

Figure 8. Receptive field of sinusoidal neural network using S.2

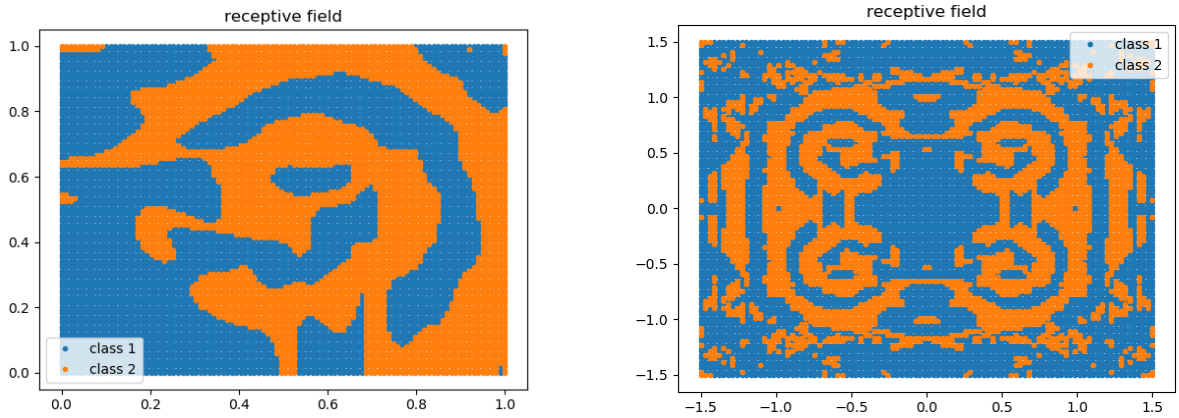

Figure 9. Receptive field of sinusoidal neural network using S.3

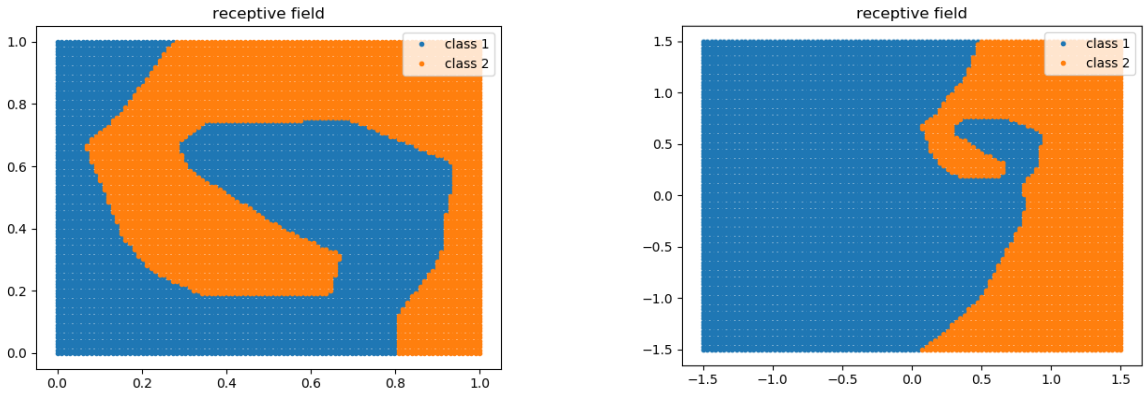

Figure 10. Receptive field of standard neural network

Figure 11 shows the receptive field of the standard neural network. The reason behind the low accuracy for the neural network can be easily understood from this receptive field demonstration. The model could not divide the field properly. This is due to the fact that the number of epochs for training the model is not sufficient. It just found a dividing line that reaches to the highest accuracy rate that it could.

5 Conclusion

During the research, biological neurons are reviewed to model a node operation for an artificial neuron. It has been seen that biological neurons are transmitting the information with signals

behaving as periodic functions. According to this outcome, the use of periodic functions in artificial neural networks (ANN) is reviewed. It is found that periodic functions are used as activation functions and as node operations. The ones which use sine as node operation are called Fourier Neural Network (FNN). Both FNN and ANN using sine as activation function have shown that the division of probabilistic space can be infinite with periodic functions. The effects of this phenomenon are experimented and proved in [9], [14]. The infinite number of divisions in probabilistic space lets the ANN train much faster than standard ANN.

The behavior of sine function in node operation is also demonstrated using unit circles, and new node operations that use sine are redesigned. Therefore, the behavior and features of sine node operation are understood. After formulas were defined, experiments were conducted to see practical results. There were no improvement in the experiment with the MNIST [20] dataset, which may be because the MNIST dataset is a dataset that is almost linearly separable. On the other hand, the experiment that was made with two spiral dataset has shown the success of sinusoidal neural networks (SNN). While S.2 has achieved the highest accuracy, S.3 showed an intriguing generalization capacity. While the variations of S.1 and S.2 are applied in previous researches, S.3 has been applied firstly in this research. The results which are obtained in the experiments are promising. Furthermore, a thorough comparison of the achieved results against ReLU, leaky ReLU and parameterized ReLU should also be investigated.

In future work, several studies on SNNs are planned to be conducted:

- A deeper investigation of SNNs with S.2 and S.3 to find methods for initial weights, which has become the biggest issue in the experiments that are applied in this research.
- Application of SNNs on more complex datasets and problem domains to assess the capacity of SNNs further.
- The use of SNNs in generative adversarial networks as the fully connected layers.

Acknowledgments

This research has been conducted during studies at the master study program Business Informatics at Riga Technical University and is a condensed version of Master Thesis. I would like to express my deepest appreciation to Dr.sc.ing. Ilze Birzniece who was my academic advisor during thesis research. I would like to extend my sincere thanks to my colleagues from Riga Liquid Studio of Accenture Latvia for their motivational support.

References

[1] D. Huang, “How much did AlphaGo-Zero cost?” Available: https://www.yuzeh.com/data/agz-cost.html

[2] F. F. Rosenblatt, “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain,” *Psychological review*, vol. 65, no. 6, pp. 386–408, 1958. Available: https://doi.org/10.1037/h0042519

[3] D. Debanne, E. Campanac, A. Bialowas, E. Carlier, and G. Alcaraz, “Axon Physiology,” *Physiological Reviews*, vol. 91, no. 2, pp. 555–602, 2011. Available: https://doi.org/10.1152/physrev.00048.2009

[4] P. A. Calabresi, “Multiple Sclerosis and Demyelinating Conditions of the Central Nervous System,” *Goldman-Cecil Medicine*. 25th ed., chap 411, Philadelphia, 2016.

[5] K. Susuki, “Myelin: A Specialized Membrane for Cell Communication,” *Nature Education*, vol. 3, no. 9, p. 59, 2010.

[6] M. W. Barnett and P. M. Larkman, “The Action Potential,” *Practical Neurology*, vol. 7, no. 3, pp. 192. PMID 17515599, 2007. Available: https://pn.bmj.com/content/7/3/192

[7] M. B. Jensen “Neuron Action Potential Mechanism,” Available: https://www.khanacademy.org/test-prep/mcat/organ-systems/neuron-membrane-potentials/v/neuron-action-potential-mechanism

[8] “Action Potential in the Neuron,” Harvard extension school, Available: https://www.youtube.com/watch?v=oa6rvUJlg7o

[9] J. M. Sopena, E. Romero, and R. Alquezar, “Neural Networks with Periodic and Monotonic Activation Functions: A Comparative Study in Classification Problems,” 1999. Available: https://doi.org/10.1049/cp:19991129

[10] G. Parascandolo, H. Huttunen, and T. Virtanen, “Taming the Waves: Sine as Activation Function in Deep Neural Networks,” 2017. Available: https://openreview.net/pdf?id=Sks3zF9eg

[11] A. Silvescu, “Fourier Neural Networks,” *IJCNN'99, International Joint Conference on Neural Networks*, vol. 1, pp. 488–491, 1999. Available: https://doi.org/10.1109/ijcnn.1999.831544

[12] H. S. Tan, “Fourier Neural Networks and Generalized Single Hidden Layer Networks in Aircraft Engine Fault Diagnostics,” *Journal of Engineering for Gas Turbines and Power*, vol. 128, no. 4, pp. 773–782, 2006. Available: https://doi.org/10.1115/1.2179465

[13] A. Zhumekenov, “Convergence Rate of Fourier Neural Networks.” Master thesis, Dept. of Mathematics, Nazarbayev University, 2019.

[14] S. Liu, “Fourier Neural Network for Machine Learning,” *2013 International Conference on Machine Learning and Cybernetics*, pp. 285–290, 2013. Available: https://doi.org/10.1109/icmlc.2013.6890482

[15] A. Zhumekenov, M. Uteuliyeva, O. Kabdolov, R. Takhanov, Z. Assylbekov, and A. J. Castro, “Fourier Neural Networks: A Comparative Study,” 2019.

[16] G. Sanderson, “But what is a Fourier Series? From heat flow to circle drawings,” Available: https://www.youtube.com/watch?v=r6sGWTCMz2k&t=4s

[17] Ch. Stover, “Unit Circle.” Available: http://mathworld.wolfram.com/UnitCircle.html

[18] Unit circle. Available: https://en.wikipedia.org/wiki/Unit_circle

[19] A.R. Gallant, H. White, “There exists a neural network that does not make avoidable mistakes,” *Proceedings of the Second Annual IEEE Conference on Neural Networks, San Diego, CA, USA*, 1988. Available: https://doi.org/10.1109/icnn.1988.23903

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. Available: https://doi.org/10.1109/5.726791

[21] T. E. Ozmermer, “Sinusoidal Neural Networks Experiments 4.1.” Available: https://github.com/evrimozmermer/sinusoidal_neural_networks_experiments_4_1