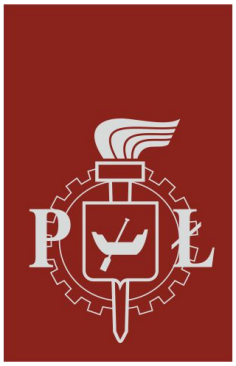

Łódź University of Technology

International Faculty of Engineering

BSc in Computer Science

Course: Numerical Methods

Lecturer: Prof. Paolo Di Barba

Academic Year: 2023-24

Understanding the Significance of Hidden Layers and Activation Functions in Artificial Neural Networks

Candidate: Jakub Dulas
Student Number: 246930

Contents

1 Introduction

1.1 Background

Artificial Neural Networks (ANNs) have emerged as powerful tools in the field of machine learning and artificial intelligence, drawing inspiration from the structure and functioning of the human brain. The fundamental concept of ANNs revolves around the idea of learning intricate patterns and relationships within data, allowing them to excel in tasks ranging from image recognition to natural language processing.

The structure of an ANN is composed of interconnected layers, each playing a unique role in the processing of information. At the heart of this architecture lie the hidden layers, which act as powerful feature extractors and enable the network to capture complex hierarchical representations within the input data. These hidden layers are pivotal in transforming raw input into meaningful and abstracted features, making them indispensable for tackling real-world challenges.

Equally significant are the activation functions, non-linear transformations applied to the output of each neuron in the network. Activation functions introduce non-linearity to the model, allowing it to learn and approximate complex mappings between inputs and outputs. The choice of activation functions has a profound impact on the network's ability to learn and generalize, influencing its capacity to solve diverse problems. [1]

1.2 Objective

The objective of this report is to investigate and analyze the significance of hidden layers and activation functions in artificial neural networks (ANNs). Building upon previous research, I aim to delve deeper into understanding their roles, functionalities, and impact on the performance of neural network models. By conducting comprehensive case studies and experiments, I seek to uncover nuanced insights that can inform the design and optimization of neural network architectures for various machine learning tasks.

2 Definitions

2.1 ANNs

Artificial neural networks (ANNs) are machine learning models inspired by the organization of biological neural networks in animal brains. ANNs consist of intercon-

nected units known as artificial neurons, resembling neurons in the brain, connected via edges simulating synapses. Each neuron receives signals from connected neurons, processes them using a non-linear activation function, and passes the output to other connected neurons. Neurons possess weights that adjust during learning, affecting signal strength. Neurons are typically organized into layers, with each layer performing specific transformations on inputs. Signals propagate from the input layer through hidden layers to the output layer. ANNs, especially those with multiple hidden layers, termed deep neural networks, are applied in predictive modeling, adaptive control, artificial intelligence, and other domains. They learn from data and can draw insights from complex information sets. [2]

A basic single hidden layer neural network can be described using the following equation:

$$y(\mathbf{x}) = \sigma_2 \left(\mathbf{W}_2 \cdot \sigma_1 \left(\mathbf{W}_1 \cdot \mathbf{x} + \mathbf{b}_1 \right) + \mathbf{b}_2 \right) \tag{1}$$

where:

$\mathbf{x}$ represents the input vector,

$\mathbf{W}_1, \mathbf{W}_2$ are the weight matrices for the first and second layers, respectively,

$\mathbf{b}_1, \mathbf{b}_2$ are the bias vectors for the first and second layers, respectively,

σ_1, σ_2 are the activation functions applied element-wise.

2.2 Hidden Layers

Hidden layers within artificial neural networks serve as intermediary layers of neurons, situated between the input and output layers. These layers are pivotal in making neural networks "deep", facilitating the learning of intricate data representations. Acting as the computational backbone of deep learning models, hidden layers enable neural networks to approximate functions and discern patterns within input data.

The primary function of hidden layers is to process inputs to a form usable by the output layer. This involves applying weights to inputs and passing them through an activation function, allowing the network to grasp non-linear relationships between input and output data.

Within a hidden layer, each neuron receives inputs from all neurons in the preceding layer, then applies weights to these inputs, adds a bias term, and applies an activation function. The resulting output of each neuron is subsequently utilized as input for the subsequent layer. [3]

2.3 Activation Functions

The activation function in an artificial neural network computes the output of a node based on its inputs and corresponding weights. The choice of a nonlinear activation function enables the network to effectively tackle complex problems, often requiring only a limited number of nodes for solution. Contemporary activation functions encompass various options, including the smooth variant of ReLU (eq. 2), known as GELU (eq. 3), notably employed in the 2018 BERT model. Additionally, the logistic (sigmoid) function, utilized in the 2012 speech recognition model by Hinton et al., and ReLU, featured in the 2012 AlexNet computer vision model and the 2015 ResNet model, are prevalent choices. [4]

$$\text{ReLU}(x) = \max(0, x) \tag{2}$$

$$\text{GELU}(x) = \frac{1}{2}\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}} \cdot \left(x + 0.044715 \cdot x^3\right)\right)\right) \tag{3}$$

3 Case studies

3.1 Varying Activation Functions and the Number of Neurons

3.1.1 Objective

The primary objective of this experiment is to investigate the impact of activation functions and the number of neurons in the hidden layer on the performance of neural networks for the task of approximating a sinusoidal wave dataset with added noise. The experiment aims to identify which combination of activation functions and hidden layer sizes leads to better generalization and accuracy in capturing the underlying pattern of the sinusoidal wave. Additionally, it seeks to compare the performance of models with different activation functions, including ReLU and GELU, to understand their respective advantages in this specific function approximation task.

3.1.2 Dataset

The dataset used for this experiment is a sinusoidal wave which is represented by the formula:

$$f(x) = \sin(4\pi x) \tag{4}$$

Additionally I added some noise to the function. The data is presented in Figure 1 and Figure 2.

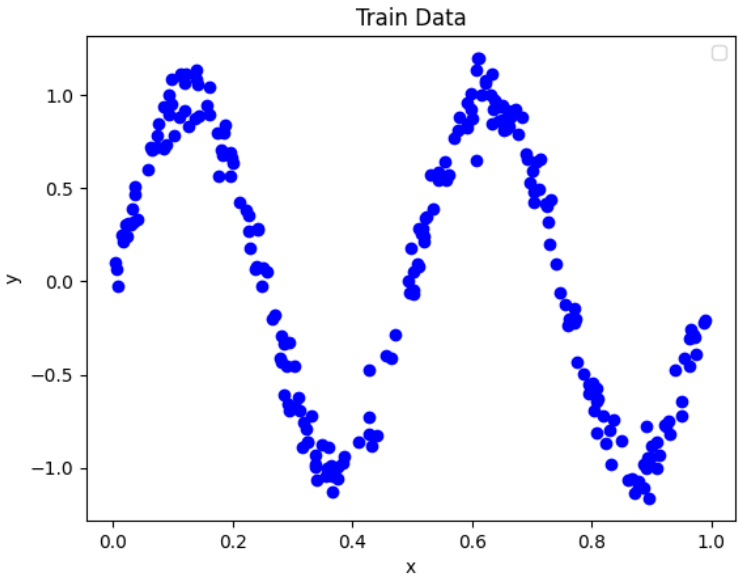

Figure 1: The figure shows data on which the neural networks are trained.

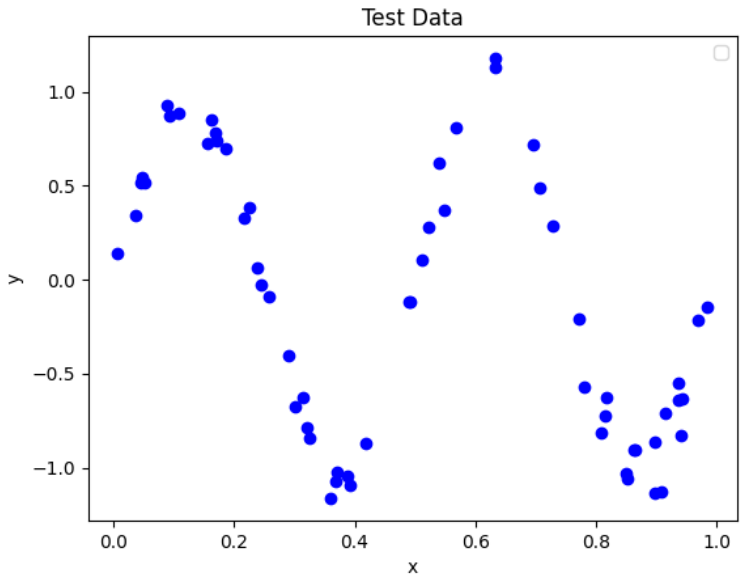

Figure 2: The figure shows data on which the neural networks are evaluated.

3.1.3 Neural Network Architectures

I constructed 24 distinct neural networks with differing quantities of neurons in the hidden layer (ranging from 2 to 256). Each model's input layer processes one feature and yields a single output for a given input. The initial group employs ReLU activation, the second group applies no activation function, and the third group utilizes the GELU activation function.

Three types of neural network architectures are considered:

1. **Model 1:** Linear model without any activation function.
2. **Model 2:** Linear model with ReLU activation function in the hidden layer.
3. **Model 3:** Linear model with GELU activation function in the hidden layer.

I adopt a specific naming convention for our neural network models. The model designation `Model_N` comprises two key elements:

- `'Model'`: Signifying a specific neural network architecture, this model configuration may or may not incorporate an activation function in the hidden layers.

- `'_N'`: This part of the name indicates the number of neurons in the hidden layer of the model. In the case of `Model_N`, the hidden layer is configured with N neurons.

3.1.4 Implementation Details

The neural networks were trained using the Mean Squared Error (MSE) loss function (5) and the Adam optimizer. The training hyperparameters include a learning rate of 0.01 and 10000 epochs.

$$\text{MSE} = (y - \hat{y})^2 \tag{5}$$

3.1.5 Results

Upon completing the training of the models, I assessed the test loss for each model, and the outcomes are displayed in Table 1. The minimum test loss for each model has been highlighted.

Model	Test Loss
model1_2	0.42663484811782837
model1_4	0.42663559317588806
model1_8	**0.4230281412601471**
model1_16	0.4266357421875
model1_32	0.42680874466896057
model1_64	0.4266357421875
model1_128	0.4266357421875
model1_256	0.4266357123851776
model2_2	0.42677953839302063
model2_4	0.3619600832462311
model2_8	0.42705127596855164
model2_16	0.09144775569438934
model2_32	0.09137598425149918
model2_64	0.0802164152264595
model2_128	0.015543906949460506
model2_256	**0.011234394274652004**
model3_2	0.24625711143016815
model3_4	**0.010816454887390137**
model3_8	0.01105255726724863
model3_16	0.010886384174227715
model3_32	0.012979873456060886
model3_64	0.011134386993944645
model3_128	0.010991798713803291
model3_256	0.011071598157286644

Table 1: Test Loss for Different Models

I generate a plot depicting the original function (eq. 4) and then proceed to compare the actual values with the output of various models.

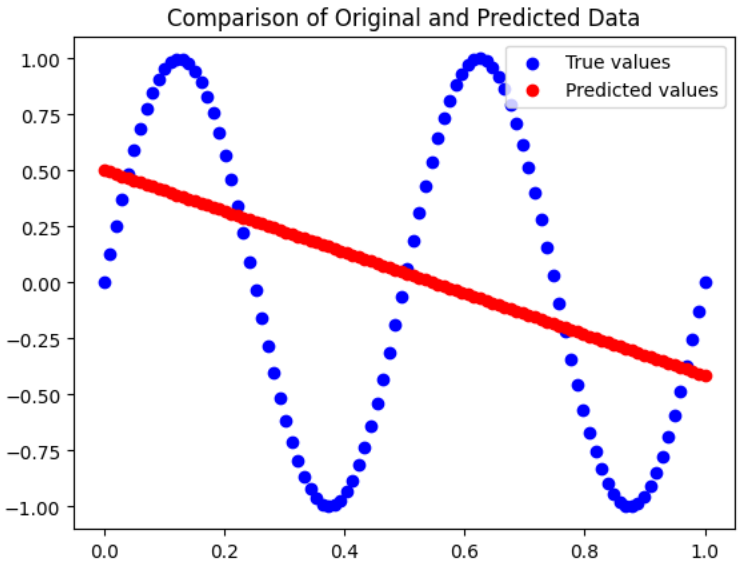

Figure 3: `model1_8` results.

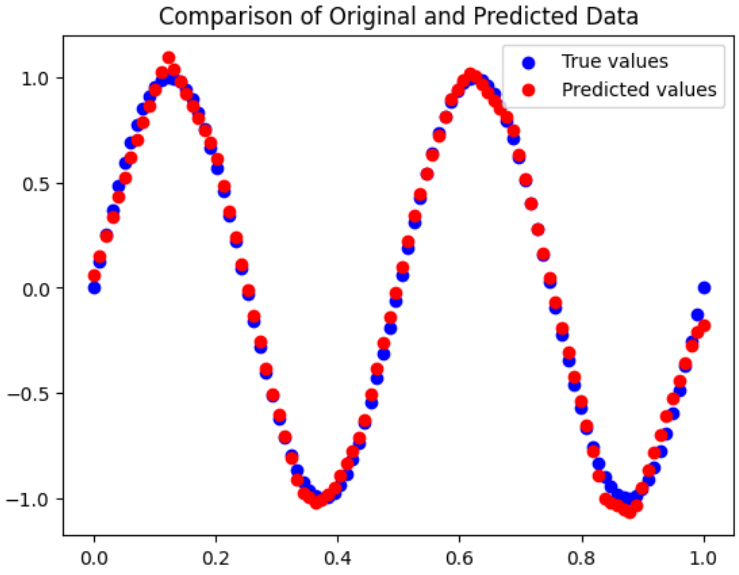

Figure 4: `model2_256` results.

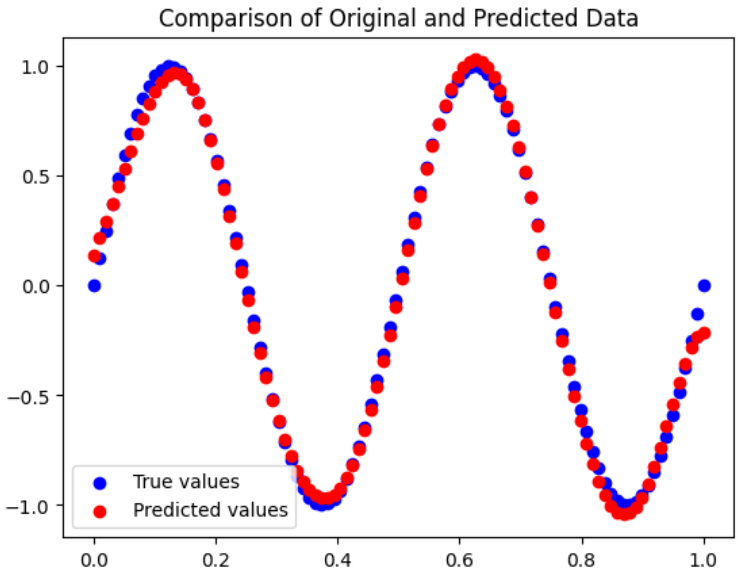

Figure 5: `model3_4` results.

As illustrated in Figure 3, introducing non-linearity to the model proves to be essential for capturing patterns. While `model1_8` successfully identifies the best-fitting line, its performance falls short when compared to the actual values for the given function.

The comparison between original and predicted data in Figure 4 demonstrates a notable alteration in the neural network's behavior due to the implementation of the ReLU activation function. Although this model can closely fit the curve, it requires a substantial number of parameters.

Among all the models, `model3_8` exhibits the lowest test loss. The utilization of the GELU activation function not only allows for a reduction in the number of parameters but also leads to improved results.

3.2 Varying Number of Layers and Number of Neurons in Hidden Layers

3.2.1 Objective

The aim of this experiment is to explore the importance of the quantity of hidden layers within an artificial neural network. Additionally, I vary the number of neurons within each hidden layer to determine the most optimal combination for our specific problem.

3.2.2 Dataset

Fashion-MNIST [5], created by Zalando, comprises a collection of images depicting various articles, with 60,000 examples in the training set and 10,000 examples in the test set. Each image is grayscale and measures 28x28 pixels, labeled into 10 distinct classes. Zalando designed Fashion-MNIST as a direct substitute for the traditional MNIST dataset, maintaining identical image dimensions and split structure, facilitating seamless benchmarking of machine learning algorithms.

Each image comprises 28 by 28 pixels, totaling 784 pixels. Each pixel is represented by a single value ranging from 0 to 255, indicating its lightness or darkness, with higher values indicating darker shades.

Each training and test example is assigned to one of the following labels:

1. T-shirt/top
2. Trouser
3. Pullover
4. Dress
5. Coat
6. Sandal
7. Shirt
8. Sneaker
9. Bag
10. Ankle boot

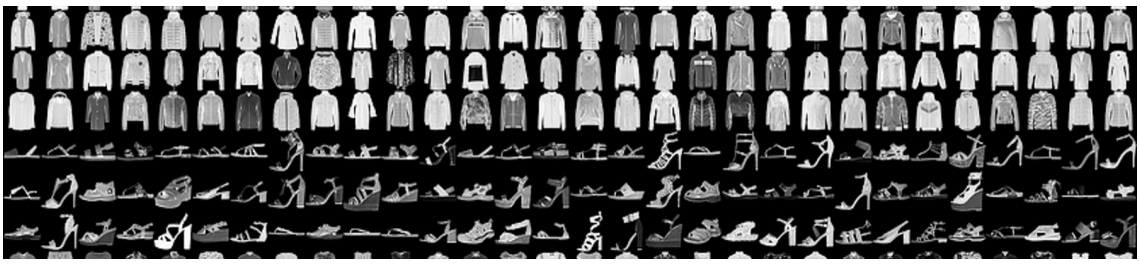

Figure 6: Fashion-MNIST

3.2.3 Neural Network Architectures

I've devised a set of 16 unique models, each tailored to handle classification tasks with 10 output classes. The input layer of each model accepts 784 features, indicating compatibility with 28x28 pixel images flattened into a vector. These models vary in complexity, ranging from architectures with no hidden layers to those incorporating up to three hidden layers. In each hidden layer, the neuron count spans from 8 to 128, ensuring a diverse range of model complexities. Notably, following each layer addition, I employed the rectified linear unit (ReLU) activation function. The naming convention employed for these models remains consistent with that utilized in the previous case study, facilitating straightforward reference and ensuring coherence across experiments.

The naming convention follows the format `modelX_Y`, where:

- `X` denotes the number of hidden layers in the neural network. It can range from zero to any positive integer.
- `Y` represents the number of neurons within each hidden layer, and it may not exist if there are no hidden layers in the model.

3.2.4 Implementation Details

The experiment entailed the training of 16 distinct neural networks. Throughout each epoch, metrics including train loss, train accuracy, test loss, and test accuracy were systematically collected. Then these metrics were utilized to generate four comprehensive graphs. Following the training process, each model underwent evaluation, with the resultant metrics saved for future comparative analysis. Categorical crossentropy loss was employed as the loss function during training. Training was conducted over 100 epochs, employing a learning rate of 0.1 and Stochastic Gradient Descent (SGD) as the optimizer.

3.2.5 Results

Upon training 16 models, I preserved the training metrics in the following table.

Model	Test Accuracy	Test Loss
model0	55.59	1.19
model1_8	87.26	0.36
model1_16	90.68	0.25
model1_32	93.64	0.17
model1_64	95.72	0.11
model1_128	**97.33**	**0.07**
model2_8	87.29	0.36
model2_16	90.69	0.25
model2_32	92.62	0.19
model2_64	95.14	0.12
model2_128	96.85	0.08
model3_8	83.77	0.46
model3_16	89.52	0.30
model3_32	91.44	0.22
model3_64	94.98	0.13
model3_128	97.28	0.07

Table 2: Test Results

Upon examining the test accuracy and loss metrics, it's evident that a model lacking hidden layers struggles to discern the patterns needed to accurately classify objects in images. The accuracy of `model0` is insufficient for practical use. Models with hidden layers, on the other hand, demonstrate significantly better performance. Interestingly, the model with the highest accuracy only employs a single hidden layer with 128 neurons, a result comparable to models with three hidden layers of the same neuron count. This suggests that for this problem, increasing model depth beyond one layer is unnecessary. While depth does play a role, the number of neurons in hidden layers appears to be the crucial factor. The analysis indicates that augmenting the number of neurons in hidden layers enables the model to better memorize patterns from the data, leading to improved predictions.

4 Conclusion

In this report, I delved into the significance of hidden layers and activation functions in artificial neural networks (ANNs), providing insights into their roles, functions, and impact on model performance. Through comprehensive case studies, I explored

various aspects of these components and their interplay, shedding light on their importance in designing effective neural network architectures.

Firstly, I investigated the impact of activation functions and the number of neurons in hidden layers on the performance of neural networks for function approximation tasks. My experiments revealed that the choice of activation function and the number of neurons significantly influence the network's ability to capture complex patterns in data. I observed that introducing non-linearity through activation functions such as ReLU and GELU is essential for enhancing model performance, with GELU showing promising results due to its smoother transition. Additionally, I found that increasing the number of neurons in hidden layers can improve model accuracy, albeit with diminishing returns beyond a certain point.

Secondly, I explored the importance of the quantity of hidden layers in neural networks using the Fashion-MNIST dataset. The experiments demonstrated that while depth plays a role in improving model performance, the number of neurons in hidden layers is a crucial factor. Surprisingly, a single hidden layer with a sufficient number of neurons can achieve comparable results to deeper architectures, suggesting that increasing model depth beyond a certain point may not always lead to significant improvements.

In conclusion, understanding the synergy between hidden layers and activation functions is crucial for designing efficient and effective neural network models. By carefully selecting appropriate activation functions and tuning the number of neurons in hidden layers, it is possible to build neural networks capable of accurately capturing complex patterns in data, leading to improved performance across various tasks in machine learning and artificial intelligence.

References

[1] Hardesty, Larry (14 April 2017). *Explained: Neural networks*. MIT News Office. Retrieved 2 June 2022.

[2] Wikipedia, ANN

[3] DeepAI, Hidden Layer

[4] Wikipedia, Activation Function

[5] Zalando Research, Chris Crawford *Fashion MNIST*.

[6] Jakub Dulas *Code to the project*.