A closer look at the impact of activation functions on Generalizability

Gabriela Tsvetkova
gabriellatsvetkova43@gmail.com

Under the guidance of Victor Kolev,
Stanford University

2020

Abstract

The aim of this research is to observe the impact of different activation functions on neural network's adaptiveness to unseen data. We do so by training neural networks to approximate fundamental mathematical functions and test them on data outside of the training domain. Empirically, it can be inferred that activation functions implement implicit structures in the network, which biases training towards functions similar to the behaviour of the aforementioned functions. Hence, activation functions provide an inductive bias that if properly understood can aid improve training stability and generalization.

Content

1 Introduction

Over the past decades, an overwhelming body of research has been constituting the effectiveness of Deep Learning. These black-box models – neural networks – are not explicitly programmed for a particular task, instead an ability to optimize their performance is implemented. They do so as they gain experience starting from zero, and gradually acquire potential for recognizing certain patterns.

However, recent applications of neural networks models proved that they are very beneficial for certain tasks, but limited for others as they perform poorly on new data. Clearly, there are a few methods for improving this feature, but as observed in (Zhang et al., 2017), explicit regularization techniques scarcely contribute to it. Therefore other factors should be taken into consideration as well.

Neural networks are in essence a composition of affine transformations. However, linearity does not give the model the capacity needed to approximate any function given unlimited data and infinite time for training. That is why non-linear activation functions such as Sigmoid, Tanh, ReLU are introduced. They actively participate in the learning process so they should have some contributions to the net's generalizability.

As we have been experimenting with the task of approximating fundamental mathematical functions such as $y = x^2$ and $y = x^3$, we noticed the following: it can be inferred that activation functions implement explicit structures in the network, which biases training towards functions similar to the behaviour of the aforementioned functions.

In this paper we examine the influence of 6 activation functions and provide empirical evidence for our hypothesis. This work paves way for research into the inductive biases of neural network design, which if properly exploited could improve network stability, training speed, generalization, and applicability.

2 Concepts

2.1 Feedforward neural network

A feedforward neural network consists of nodes and connections between them. Each neuron is associated with an activation and a bias and each connection – with a certain weight.

The value of the activation function of the neurons in layer l is obtained by the following formula:

$$\boldsymbol{a}^{(l)} = g(\boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)})$$

Here, n_l is the number of units in layer l; $\boldsymbol{a}^{(l)} \in \mathbb{R}^{n_l}$ is the vector containing the activations of the neurons in layer l; $\boldsymbol{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ is the matrix containing the weights which connect the neurons in layer l to those in layer $l-1$;

$\boldsymbol{b}^{(l)} \in \mathbb{R}^{n_l}$ is the vector containing the biases of the neurons in layer l; and g is some non-linear activation function.

2.2 Loss function

The loss function evaluates how well specific algorithm models the given data i.e. the error. Since we are working in a continuous domain, we use the mean square error function:

$$\mathcal{L} = \frac{1}{n_L} \|\boldsymbol{y} - \boldsymbol{a}^{(L)}\|^2$$

Here, $\boldsymbol{y} \in \mathbb{R}^{n_L}$ is the vector containing the ground truth outputs; $\boldsymbol{a}^{(L)} \in \mathbb{R}^{n_L}$ is the vector containing the activations of the neurons in the last layer (L) i.e. the vector containing the predictions; and $\|.\|$ is the Euclidean norm. Two algorithms are involved in the process of learning of the neural network – backpropagation and gradient descent. For the latter we use Adam [2] in our experiments.

3 Methods

3.1 Activation Functions

In this paper the activation functions used are as follows:

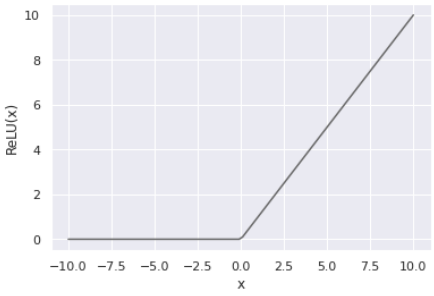

Figure 1: $\text{ReLU}(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$

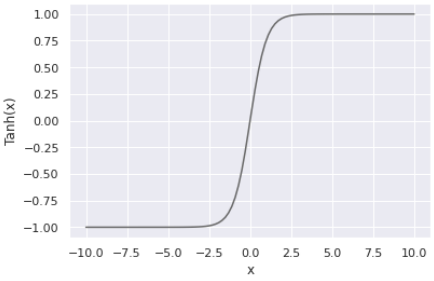

Figure 2: $\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

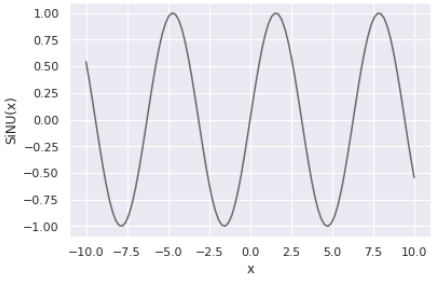

Figure 3: $\text{SiNU}(x) = \sin(x)$

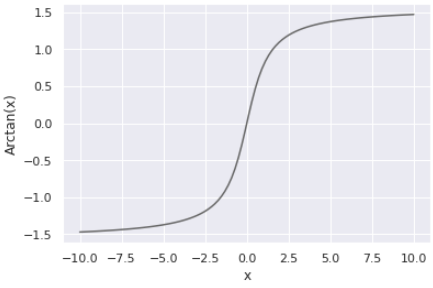

Figure 4: $\text{Arctan}(x) = \tan^{-1}(x)$

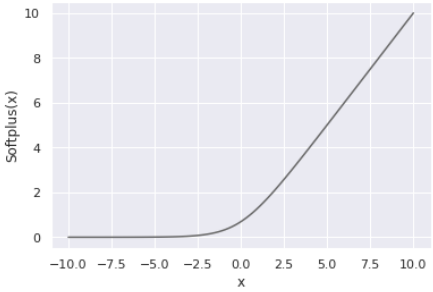

Figure 5: $\text{Softplus}(x) = \ln(1 + e^x)$

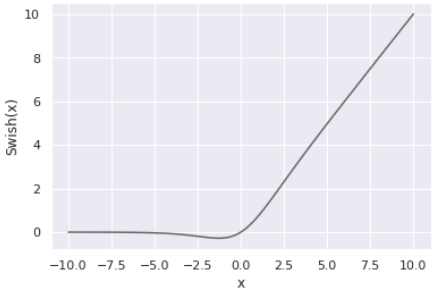

Figure 6: $\text{Swish}(x) = \frac{x}{1+e^{-x}}$

3.2 Spectral Norm Regularization

Neural networks tend to learn the training examples well, but fail to generalize to new ones. This is known as overfitting, and stems from the fact that the

minimum error objective is ill-posed. For instance, the Lagrange polynomial of a set of points would have the smallest loss, yet it would not have high generalization error.

One method of alleviating overfitting is regularization. This is the process of penalizing big values in the weight matrix with the effect of deriving a model which adapts well to new data

In this work we focus on spectral norm regularization[5]. It is explicitly shown that this method contributes the most to the generalizability of the model from other existing methods such as Weight Decay, Dropout, Jacobian regularization, just to name a few. (Neyshaubar etal 2019)[3] and [1]. The general idea behind it is to make the model insensitive to small input perturbation, which is central for its adaptiveness to new examples. This is achieved by using the spectral norm or in other words, the largest single value of a matrix $\boldsymbol{W} \in \mathbb{R}^{m \times n}$:

$$\sigma = \max_{\xi \in \mathbb{R}^n, \xi \neq 0} \frac{\|A\xi\|_2}{\|\xi\|_2}$$

Then for a small perturbation $\boldsymbol{\xi} \in \mathbb{R}$ where f_θ is the objective function and $\boldsymbol{W}_{\Theta,x} \in \mathbb{R}^{n_0 \times n_L}$, $\boldsymbol{b}_{\Theta,x} \in \mathbb{R}^{n_L}$ are weight matrix and bias vector respectively:

$$\frac{\|f_\Theta(\boldsymbol{x}+\boldsymbol{\xi}) - f(\boldsymbol{x})\|_2}{\|\boldsymbol{\xi}\|_2} = \frac{\|(W_{\Theta,\boldsymbol{x}}(\boldsymbol{x}+\boldsymbol{\xi}) + \boldsymbol{b}_{\Theta,\boldsymbol{x}}) - (W_{\Theta,\boldsymbol{x}}\boldsymbol{x} + \boldsymbol{b}_{\Theta,\boldsymbol{x}})\|_2}{\|\boldsymbol{\xi}\|_2} =$$
$$= \frac{\|W_{\Theta,x}\boldsymbol{\xi}\|_2}{\|\boldsymbol{\xi}\|_2} \leq \sigma\left(W_{\Theta,\boldsymbol{x}}\right)$$

3.3 Approach

Our approach is to train a neural model to approximate $y = x^2$ or $y = x^3$ only with the help of spectral norm regularization on the interval $[-15; 15]$. Then we see how the model generalize on the interval $[-40; 40]$. All the networks have a dense-layered architecture and the maximum units of a layer L are no more than 256.

4 Results

In the graphs below are illustrated the models with the best performance on the two tasks.

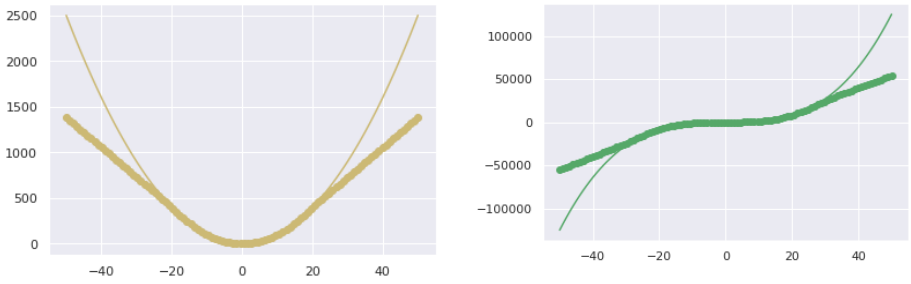

Figure 7: The left graph shows a model that approximates $y = x^2$ trained on range [-15;15] with ReLU. It is visible that the graph of the objective function keeps a linear structure because of its activation function. The same can be inferred for the right graph of $y = x^3$.

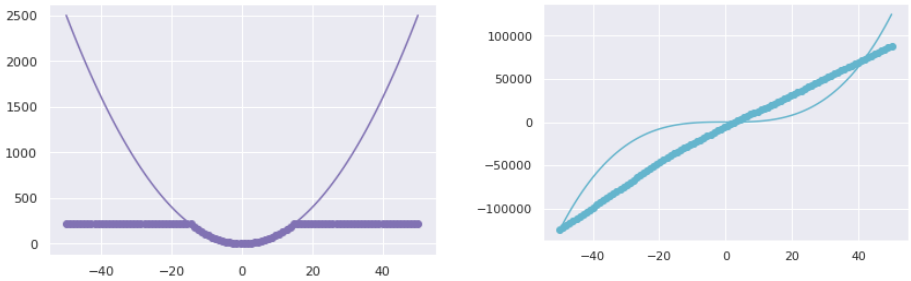

Figure 8: The left graph shows a model that approximates $y = x^2$ trained on range [-15;15] with Tanh. It is visible that the graph of the objective function keeps a linear structure because of its activation function. The same can be inferred for the right graph of $y = x^3$.

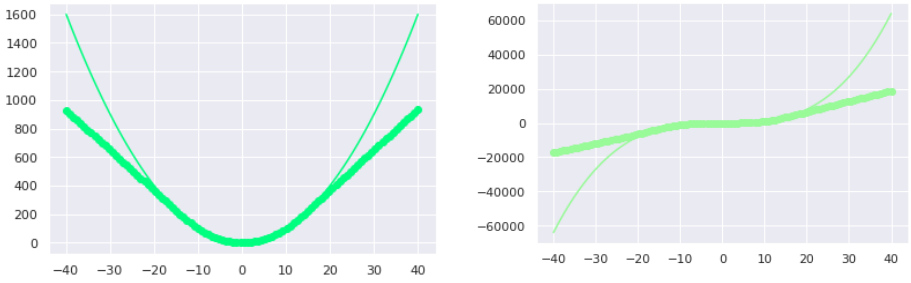

Figure 9: The left graph shows a model that approximates $y = x^2$ trained on range [-15;15] with Arctan. It is visible that the graph of the objective function keeps a linear structure because of its activation function. The same can be inferred for the right graph of $y = x^3$.

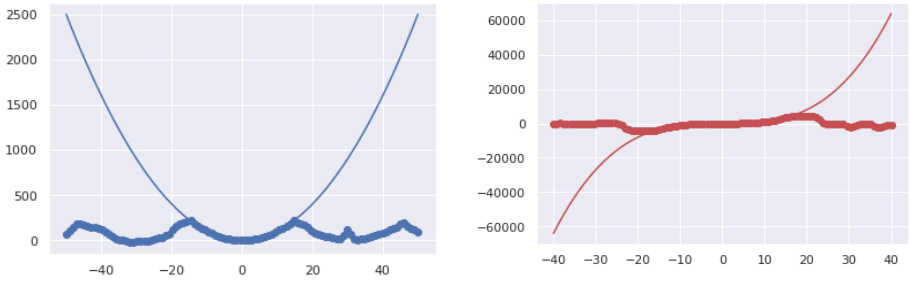

Figure 10: The left graph shows a model that approximates $y = x^2$ trained on range [-15;15] with SiNU. It is visible that the graph of the objective function keeps a sinusoidal structure because of its activation function. The same can be inferred for the right graph of $y = x^3$.

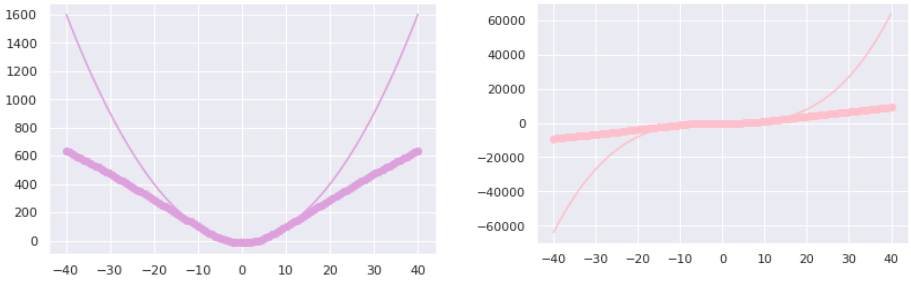

Figure 11: The left graph shows a model that approximates $y = x^2$ trained on range [-15;15] with Swish. It is visible that the graph of the objective function keeps a linear structure because of its activation function. The same can be inferred for the right graph of $y = x^3$.

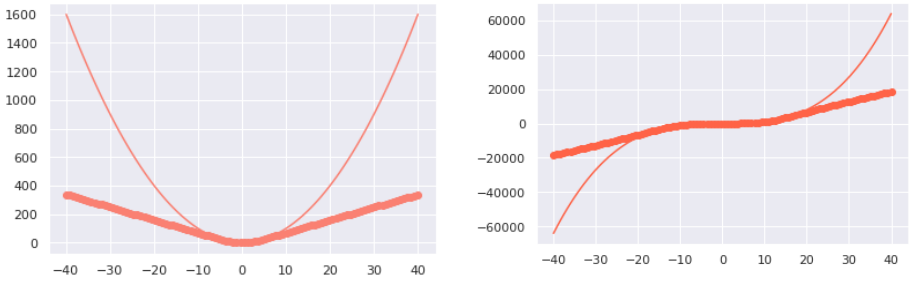

Figure 12: The left graph shows a model that approximates $y = x^2$ trained on range [-15;15] with Softplus. It is visible that the graph of the objective function keeps a linear structure because of its activation function. The same can be inferred for the right graph of $y = x^3$.

5 Future Development

Experiments such as changing the range of input data, type of function, the architecture of the neural network and the type of regularization Dropout[4] are always interesting and may lead to better understanding of the impact of activation functions. A further investigation with modular neural networks [6] for composing complex functions can be made.

6 Conclusion

In the present work, we showed empirically that the choice of activation functions has an impact on what the neural network learns, and it provides a strong bias, especially in points outside of the training domain. Such an inductive bias may be exploited by machine learning engineers to ease training, improve stability and aid generalization performance by embedding implicit structure in the network.

References

[1] Sanjeev Arora et al. "Stronger generalization bounds for deep nets via a compression approach". In: *CoRR* abs/1802.05296 (2018). arXiv: 1802.05296. URL: http://arxiv.org/abs/1802.05296.

[2] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[3] Behnam Neyshabur et al. "Exploring Generalization in Deep Learning". In: *CoRR* abs/1706.08947 (2017). arXiv: 1706.08947. URL: http://arxiv.org/abs/1706.08947.

[4] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[5] Yuichi Yoshida and Takeru Miyato. *Spectral Norm Regularization for Improving the Generalizability of Deep Learning*. 2017. arXiv: 1705.10941 [stat.ML].

[6] Jie Zhou et al. "Graph Neural Networks: A Review of Methods and Applications". In: *CoRR* abs/1812.08434 (2018). arXiv: 1812.08434. URL: http://arxiv.org/abs/1812.08434.