Aug 10th, 12:00 AM

Analyzing the Impacts of Activation Functions on the Performance of Convolutional Neural Network Models

Gerald Onwujekwegn
Virginia Commonwealth University, onwujekwegn@mymail.vcu.edu

Victoria Y Yoon
Virginia Commonwealth University, vyyoon@vcu.edu

Activation Functions and their Impact on the Training and Performance of Convolutional Neural Network Models

Emergent Research Forum (ERF)

Gerald Onwujekwe
Department of Information Systems
Virginia Commonwealth University
Richmond, VA, USA
onwujekwegn@vcu.edu

Victoria Yoon
Department of Information Systems
Virginia Commonwealth University
Richmond, VA, USA
vyyoon@vcu.edu

Abstract

Activation functions are a very crucial part of convolutional neural networks (CNN) because to a very large extent, they determine the performance of the CNN model. Various activation functions have been developed over the years and the choice of activation function to use in a given model is usually a matter of trial and error. In this paper, we evaluate some of the most-used activation functions and how they impact the time to train a CNN model and the performance of the model. We make recommendations for the best activation functions to use based on the results of our experiment.

Keywords

Activation function, convolutional neural network, performance, accuracy, training time.

Introduction

Convolutional Neural Networks (CNN) have non-linear activation functions that allow the network to learn. This function controls how a neuron fires depending on its input received from the previous layer. There are several types of activation functions that have been used throughout the history of neural networks. The classical activation functions include the *step* function, the *sigmoid* function, and the *tanh* function. The smoothness of the sigmoid function makes it easier to devise learning algorithms (Rosebrock, 2017). However the standard activation functions, such as the sigmoid function or the tangent hyperbolic function, are contractive almost everywhere, and the gradients at the large values become almost zero which makes the updates by the stochastic gradient descent very small (Ide & Kurita, 2017). This problem is referred to in the literature as the vanishing gradient problem (Hochreiter, 1998). The vanishing gradient implies that the model can neither converge nor learn from data and therefore cannot perform its purposes such as classification or prediction. The more recent functions such as the RELUs (Hahnloser et al., 2000) and Leaky RELUs (Maas, Hannun, & Ng, 2013) avoid the vanishing gradient problem because, for a positive output, the gradient is constant and does not vanish.

Designing activation functions that enable fast training of accurate deep neural networks is an active area of research (Agostinelli et al., 2014). Researchers generally agree that the activation function employed in a CNN network impacts the accuracy of classification and time to train the network. However, this area of research is not being explored rigorously despite its potential impact on CNN performance. To fill this gap in the literature, we investigate the impact of activation functions on the prediction accuracy of the CNN model using CIFAR-10, dataset. We build a convolutional neural network and activate each layer of the CNN with various activation functions.

The remainder of this paper is organized as follows. Section 2 provides the literature review, and Section 3 presents an overview of various activation functions. Section 4 presents the architecture of CNN followed

by the five experiments that we conduct to analyze the impacts of activation functions on various aspects. Finally, section 6 offers conclusions and future research directions.

Literature Review

An approach to improve the learning of the deep neural networks is to modify the activation function of the hidden neuron (Ide & Kurita, 2017). Essentially, they are non-linear mappings interleaved at the end of a layer (Scardapane et al., 2019). Krizhevsky et al. (2012) created a convolutional neural network model that is popularly known as AlexNet that won the LSVRC 2012 contest. To make training the network faster, the CNN researchers used non-saturating neurons. They argue that the saturating nonlinear model of neuron's output is much slower than the non-saturating counterpart, referring to neuron with this nonlinearity as Rectified Linear Units (ReLUs). Deep convolutional neural networks with ReLUs train several times faster than their equivalents with tanh units(Krizhevsky et al., 2012). This statement is what leads us to investigate in quantifiable terms how the choice of neuron activation function might impact training times and the accuracy of a convolutional neural network model. In other to further improve the accuracy of the model, Krizhevsky et al. applied a technique called local response normalization which implements a form of lateral inhibition inspired by the type found in real neurons, creating competition for big activities among neurons outputs computed using different kernels(Krizhevsky et al., 2012). They applied this normalization after applying the ReLU nonlinearity in certain layers of the network and found that it improved their network's accuracy.

Goodfellow et al., (2013) developed a new type of activation function, called maxout unit, and they describe the maxout model as "simply a feed-forward architecture, such as a multilayer perceptron or deep convolutional neural network, which uses a new type of activation function - the maxout unit." Maxout networks learn not just the relationship between hidden units, but also the activation function of each hidden unit. The researchers concede that maxout features are a significant departure from the common characteristics found in other activation functions. For example, maxout is linear at every point while most other activation functions do have some curvature to them. While some are curve-linear, others are almost entirely a curve. Yet Given these departures from standard practice, it may seem surprising that maxout activation functions work at all.

Agostinelli et al., (2014) proposed an adaptive activation function called adaptive piecewise units (APL). The APL is a parametrized, piecewise linear activation function and is learned independently for each neuron using gradient descent, and can represent both convex and non-convex functions of the input. When testing their adaptive piecewise unit, they used a network that is capable of learning diverse activation functions and they suggest that the standard one-activation-function-fit-all may be less than optimal for convolutional neural networks. Kheradpisheh, Ganjtabesh, Thorpe, & Masquelier, (2018) used a step function in all convolutional layers of their network. The neurons are non-leaky, integrate-and-fire neurons, which gather input spikes from presynaptic neurons and emit a spike when their internal potentials reach a prespecified threshold (Kheradpisheh et al., 2018). Their proposed solution uses a spiking deep neural network (SDNN) and can perform well on small and medium datasets. New activation functions have been proposed such as KAFs (Scardapane et al., 2019), SLAFs (Goyal et al., 2020) and adaptive activation functions (Jagtap et al., 2020), however, none of these has become widely used in building convolutional neural network models.

Despite many prior studies, researchers still are not very certain about the activation function that works best for a given network and it is usually a case of trial and error. This trial and error method could be very expensive for large networks that take long periods to train, more so when the project is under time-constraints. To fill in this gap in the literature, we evaluate the activation functions across various types of deep learning models such as CNNs and provide insights on the impact of activation functions on the prediction accuracy and training time.

CNN Architecture

The network architecture is shown in Figure 1. It has five layers in total. We vary the activation function each time we train the network to obtain training times and accuracy reports for each activation function.

The input is a 32 X 32 RGB image and we feed this input to the first layer consisting of 96 filters and 3X3 kernel convolutions. We applied zero padding at each layer to keep the spatial outputs of the convolutions consistent with the input dimensions and to prevent the input volume from attenuating. Attenuation will in effect prevent our network from learning useful patterns.

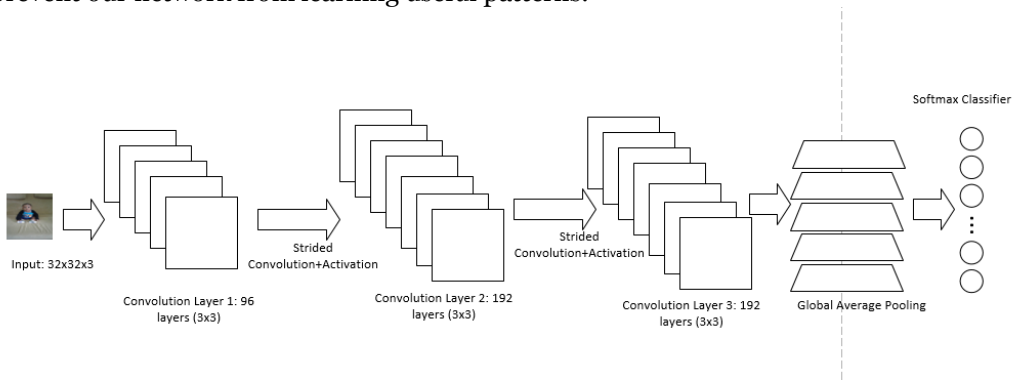


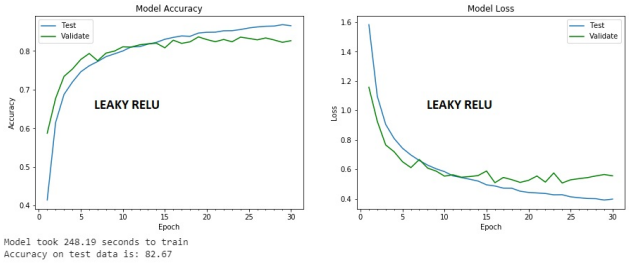
Figure 1. The CNN architecture

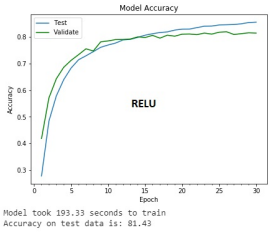
Experiment

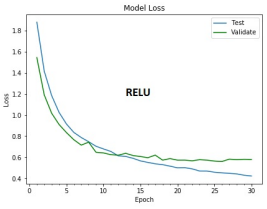
We set up an experiment to quantify the impact of activation functions on training times and the accuracy of the predictions. We used CIFAR-10 as our experiment dataset. The CIFAR-10 dataset developed by (Krizhevsky & Hinton, 2009) consists of sixty thousand (60,000) color images which are 32x32 in size and consist of 10 classes. The dataset is completely balanced with each class having 6000 images. There are 50,000 training images and 10,000 test images. Our hardware is a single NVidia Tesla K80 GPU card with 13GB of RAM and our disk size is 400GB. We ran the entire model on a GPU only. We train and test our model using a given activation function and we swap another activation function after we have reset the runtime. We must reset the runtime after each run to avoid treatment diffusion that could threaten the validity of our results. We used a 3x3 strided convolution with stride 2 in our initial run; however, the training times were not significantly different while we varied the activation functions so we decided to use stride 1. We kept the epoch to a minimum ranging from 20 to 35. Higher epochs will typically lead to higher accuracy in the learning and prediction, however in this experiment, our aim is not to optimize the model for highest accuracy digits, and hence we did not set high numbers for training epochs.

Results:

The results of our experiment are presented in the line graphs below. We used five different activation functions for our experiment; the LeakyRELU, PRELU, RELU, Tanh and Sigmoid. These are some of the popular activation function typically applied in neural network architectures. Based on current best practices, we used a stochastic gradient descent (SGD) for updating the weight matrix and computation of the gradient. We used a set momentum of 0.9 for the entire training process.







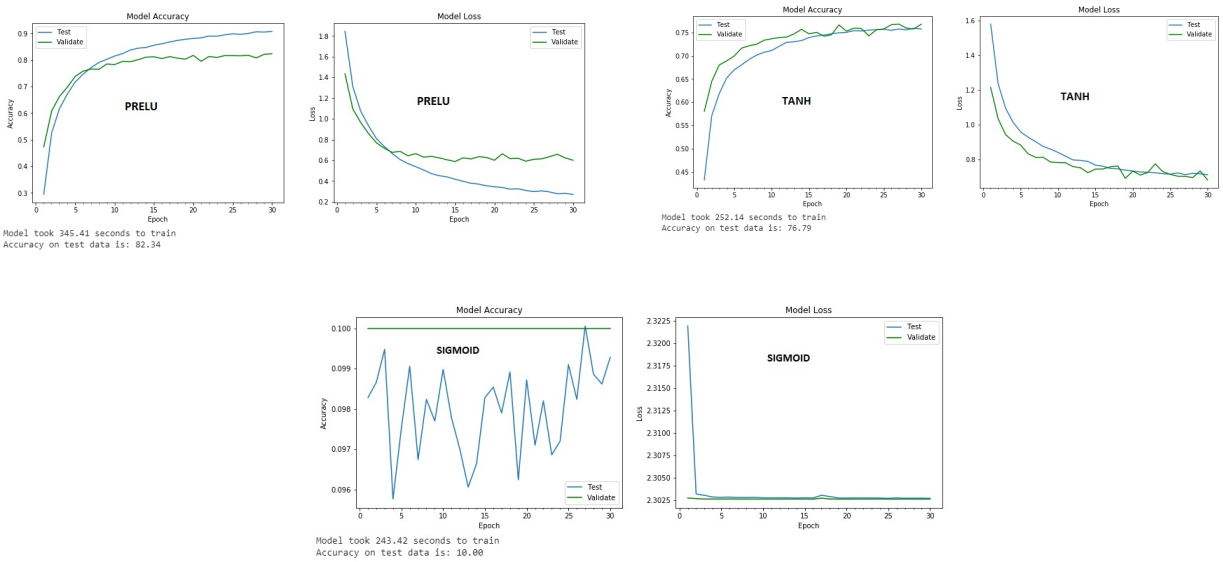


Figure 2. Accuracy and Loss Plot of the Network with Five Activation Functions

The accuracy and loss plot in figure 2 show the network performance over the training epochs. Results about the accuracy are discussed in figure 3, however, we note that the loss, which is a prediction error that is made as the model iterates through the dataset, is expected to keep decreasing as the model learns. The lower the loss values, the better the model and we have the lowest loss values on the RELU and the highest loss on the Sigmoid. For most activation functions, we observe a fairly consistent increase in the level of accuracy achieved with increase in epochs however, the sigmoid function produced a spiky accuracy plot indicating that it was not ideal to be used for the network.

The result in Figure 3 shows a fairly wide range of training times across the five activation functions investigated in this study. It takes between 193.3 – 345.4 seconds to train the 50,000 training samples of the CIFAR-10 dataset. As highlighted in a previous section, we find that the RELU activation trains the fastest on the CIFAR-10 dataset and this is mainly due to its computational efficiency. On the accuracy side, it was outperformed by the LeakyRELU and the PRELU on the CIFAR-10 but the gap is not very significant. Though the PRELU did better than the RELU on the CIFAR-10 dataset, it took nearly twice as much time to train.

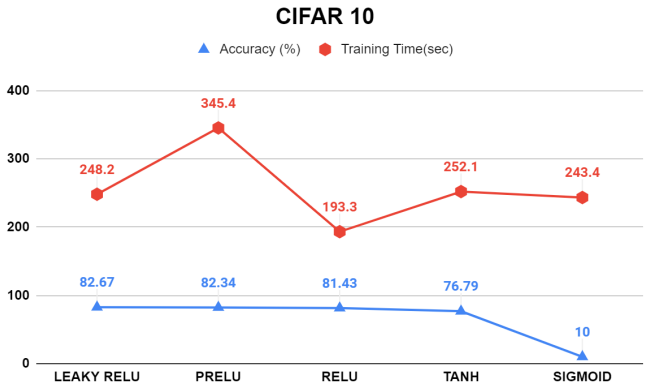


Figure 3. Impacts of Five Activation Functions on CNN Accuracy and Training Time

Conclusion

Large CNN models take a significant amount of time to train. For instance, on the ImageNet dataset, Springenberg et al., (2014) network takes four days to train; AlexNet (Krizhevsky et al., 2012) takes between

five and six days to train; ZFNet (Zeiler & Fergus, 2014) takes twelve days to train; and VGGNet (Simonyan & Zisserman, 2014) takes about two to three weeks to train. These are official numbers posted on the cited papers. A researcher trying to build the VGGNet and train it on the ImageNet dataset will have to use between 14-21 days for each given activation function if the best function for the network is unknown. To assist the researchers in choosing the best activation function, this study analyzes the impacts of five extensively used activation functions on the training time and accuracy of a CNN model. The results show that the rectified linear unit (RELU) activation function trains a CNN model quicker than any other activation function. The Leaky RELU and the PRELU did outperform the RELU on the CIFAR-10; however, there is significant amount of time involved in training the network with these activations. If every decimal digit of the achieved accuracy is important and there is sufficient time, we recommend using the Leaky RELU. Our future work will include other activation functions and quantify their impacts to make an informed choice. We will also be testing other deep learning models such as recurrent neural networks and generative adversarial networks (GANS).

References

Agostinelli, F., Hoffman, M., Sadowski, P., and Baldi, P. 2014. "Learning Activation Functions to Improve Deep Neural Networks," *ArXiv:1412.6830 [Cs, Stat]*. http://arxiv.org/abs/1412.6830

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. 2013. "Maxout Networks," *ArXiv E-Prints*, *1302*, arXiv:1302.4389.

Goyal, M., Goyal, R., Venkatappa Reddy, P., and Lall, B. 2020. "Activation Functions." In *Deep Learning: Algorithms and Applications,* W. Pedrycz and S.M. Chen (eds.), Springer, pp. 1–30.

Hahnloser, R. H. R., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. 2000. "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature* (*405*), pp. 947-951.

Hochreiter, S. 1998. "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6*(02), pp. 107–116.

Ide, H., and Kurita, T. 2017. "Improvement of learning for CNN with ReLU activation by sparse regularization," In *Proceedings of 2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2684–2691. https://doi.org/10.1109/IJCNN.2017.7966185

Jagtap, A. D., Kawaguchi, K., and Karniadakis, G. E. 2020. "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks," *Journal of Computational Physics* (404:1), 109136. https://doi.org/10.1016/j.jcp.2019.109136

Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J., and Masquelier, T. 2018. "STDP-based spiking deep convolutional neural networks for object recognition," *Neural Networks* (99), pp. 56–67. https://doi.org/10.1016/j.neunet.2017.12.005

Krizhevsky, A., and Hinton, G. 2009. "Learning multiple layers of features from tiny images," https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

Krizhevsky, A., Sutskever, I., and Hinton, G. E. 2012. "ImageNet Classification with Deep Convolutional Neural Networks," https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Maas, A. L., Hannun, A. Y., and Ng, A. Y. 2013. "Rectifier Nonlinearities Improve Neural Network Acoustic Models," In *Proceedings of the 30th International Conference on Machine Learning*.

Rosebrock, A. 2017. *Deep Learning for Computer Vision with Python*. PyImageSearch.

Scardapane, S., Van Vaerenbergh, S., Totaro, S., and Uncini, A. 2019. "Kafnets: Kernel-based non-parametric activation functions for neural networks," *Neural Networks* (*110)*, pp. 19–32. https://doi.org/10.1016/j.neunet.2018.11.002

Simonyan, K., and Zisserman, A. 2014. "Very Deep Convolutional Networks for Large-Scale Image Recognition," *ArXiv:1409.1556 [Cs]*. http://arxiv.org/abs/1409.1556

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. 2014. "Striving for Simplicity: The All Convolutional Net," *ArXiv:1412.6806 [Cs]*. http://arxiv.org/abs/1412.6806

Zeiler, M. D., and Fergus, R. 2014. "Visualizing and Understanding Convolutional Networks," In *Computer Vision – ECCV 2014* (8689), D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars (eds.), Springer, pp. 818–833).