Neural Network Expressivity

Nick Rezaee

March 2020

Abstract

A major problem in neural networks is understanding the expressivity, which is defined as "how the architectural properties of a neural network (depth, width, layer type) affect the resulting functions it can compute, and its ensuing performance." [1] Tuning neural networks to give the optimal result is very important for modern day problems and knowing how to measure a change in a neural network can be very useful. The authors of *On the Expressive Power of Deep Neural Networks* tackle this problem by contributing two new measures of expressivity and then proposing a new method to control for expressivity.

1 Introduction

Deep learning has taken the world by storm in the 21st century with applications from image recognition [2] to time series analysis [3]. Neural networks have proven themselves to be extremely useful in these cases, and there is no debate that it is a very popular method. Although it is a prominent method, it is still a fairly new prediction model. The mathematics behind neural networks is still a developing field and finding ways to understand why neural networks give the outputs they give is extremely important to every field that uses them. *On the Expressive Power of Deep Neural Networks* addresses the issue of measuring how the output of a neural network is changed from a small alteration in a parameter or input. Tuning parameters and finding how sensitive a neural network is to an input can be used to create the best neural network for a given problem. The authors of this paper propose two new methods to measure expressivity, *transitions* and *trajectories*, and create a method called *trajectory regularization* to lower the effect of a small change in the inputs on the outputs. They then compare that method to one called batch normalization to see which one performs better.

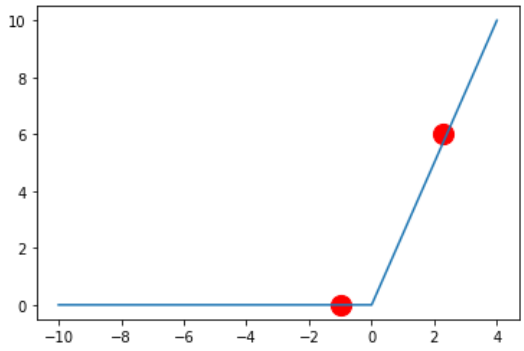
2 Expressivity Measures

In order to measure expressivity the authors give two methods. They first give one with *transitions* where the location of the input into an activation function is used. The second uses the new concept of a *trajectory length* to determine how the values change from layer to layer.

2.1 Transitions

By having an activation function with piecewise linear regions, one can measure how the inputs to the activation function are altered for a small change in the parameters and thereby see the expressivity of the neural network. A transition can have a significant effect in a ReLu activation function, where a linear region only outputs a zero value and the following linear region takes the form of $y = ax$ for some positive a. An example of a ReLu function is shown in Figure 1.

Figure 1: Relu Function



The figure describes how a ReLu function can heavily determine the output of a neuron if the point switches linear regions. In similar definition, the transition above is turning the neuron from on to off.

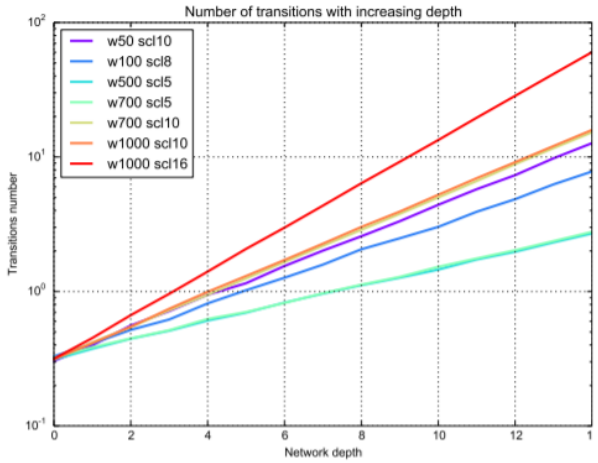
With the transitions, the authors define $T(F_A(x; W))$ to be the number of output neurons that have a transition across their activation functions with a small change in the inputs or parameters. Clearly, the maximum number $T(F_A(x; W))$ can have is the number of output neurons of the neural network. Whether or not the output neurons fire on a ReLu or what linear region the input of the activation function is computed on clearly has a large effect of what the final output of a neural network will be. By measuring these changes, we can find the differences of the overall output for a small change in the neural network.

In order to get a more holistic review of a neural network aside from just the output neurons, he defines the activation pattern as $AP(F_A(x; W))$, which is the number of transitions in the whole neural network.

This measure of expressivity gives more information about the model than just the outputs and includes the hidden layers as well. The maximum of this value is the number of neurons in the whole neural network.

The authors create neural networks and slightly change an input and see how many transitions occur from layer to layer. They then plot the number of transitions across the depth of a neural network. From Figure 2, we see that the activation pattern increases exponentially from layer to layer. The y axis is on a log scale, so we can see the exponential increase with a straight line.

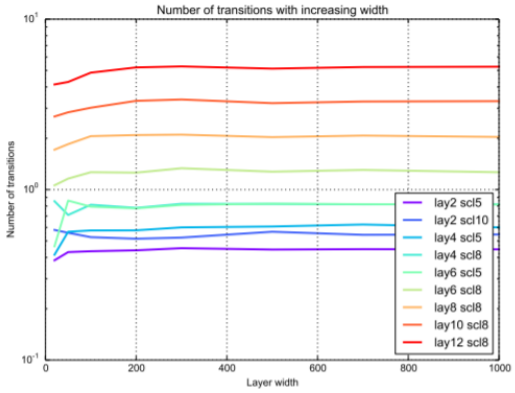
Figure 2



With seven different neural networks, there is an increase in transitions for each as the depth increases.

They also plot the number of transitions with increasing width of different types of neural network in Figure 3. Up to about 200 neurons in width, the number of transitions seems to increase, but after 200 the number of transitions seems to plateau. These results show that expressivity is much more dependent on depth than width.

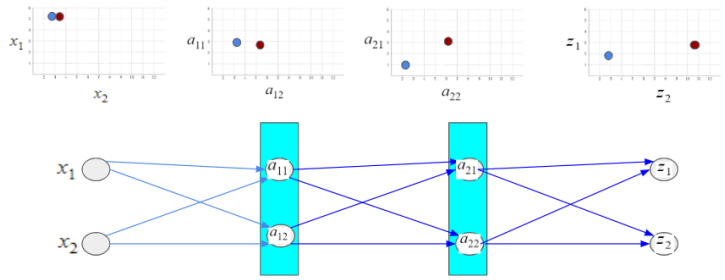
Figure 3



2.2 Trajectories

When one is trying to measure expressivity in terms of the input, one can see how the inputs compare to the outputs of the neurons of the next layer. The authors provide a more holistic method that allows one to see the changes over all the values of a curve. They show the difference between different inputs with something called a *trajectory length*. The intuition for this idea comes in Figure 4, where from the start we have two inputs that are very close to each other, but as we go propagate through the neural network we start to see that small difference in the inputs create a very large difference for future layers.

Figure 4: Expressivity with Points



In this case, the brown and blue points are very close to each other, but we can see that from layer to layer the values seem to deviate more and more.

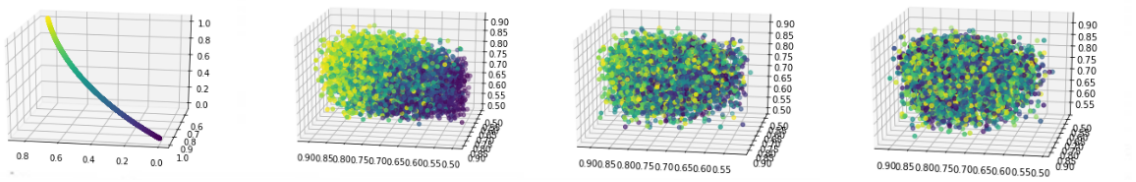
Instead of just taking two points, we can take a line with a large number of points and see how that line changes from layer to layer. The authors create a line with a parametric curve to display how a small change in the input can greatly affect the outputs of each layer. The authors simulate the changes by first inputting points of the unit circle. The authors then connect the points that are outputted from the layer to get a new trajectory. The authors' results are seen in Figure 5.

Figure 5: Expressivity With a Parametric Curve

As the points on the unit circle move from layer to layer, we can see that as the points that were next to each other are connected by a line, that the length of the curve (*trajectory length*) increases from layer to layer, signifying the increasing expressivity.

I simulate a similar example in 3D with three inputs instead of two. I color coded the inputs to stay the same throughout each layer, so one will be able to notice how far a point has gotten from its original place by noticing if the colors are as close as they were before to their original starting point. Also, the clustering of colors means that points that were initially near each other stay near each other. If the colors do not cluster overtime that means that the small changes in the input have a large effect on the outputs.

Figure 6: Reproducing Expressivity from Layer to Layer

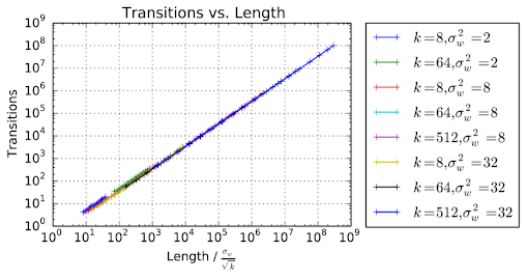


One can see that after the first hidden layer that the line becomes a mass of points, and that the colors seem to cluster in the first two hidden layers, but in the fourth layer there is clearly no clustering, and the colors seem very random.

2.3 Comparing Measures of Expressivity

If both measures of expressivity actually work, then the transitions and trajectory length should have a strong positive correlation with each other. In order to see if the measures actually do exhibit strong positive correlations for a variety of neural networks, the authors measure the expressivity of eight different neural network. In Figure 6, we do see the clear positive correlations, and not only that but we also see how deeper neural networks and models with larger variances for the weights actually have more expressivity than their counterparts. These results display that these new measures have merit.

Figure 7: Caption



In this figure, k represents the number of layers, σ_w^2 represents the variances of the weights. We expect that as the number of layers increase that so do the expressivity measures and that is the case in the graph. Also, as the variances of the weights increase, we again do see the increase in expressivity.

3 Regularization Methods

3.1 Trajectory Regularization

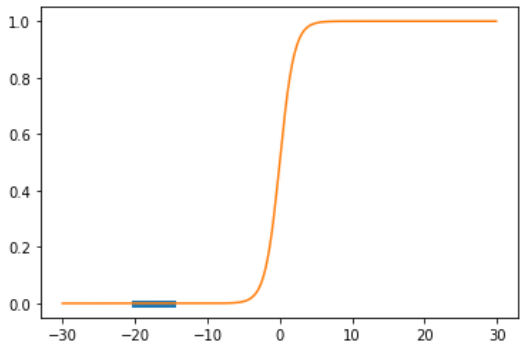
If one wants to regulate expressivity, one should create a measurement for expressivity and find a way to regulate it. That is exactly what the authors of this paper do. In this case, they regulate the length of a trajectory by including it in the loss. The authors add $\lambda(currentlayerlength/originallayerlength)$ into the loss function. One can see that it is a ratio that compares the original length and the current length. With this new loss, if the trajectory length increases the loss function will hold a higher strictness and control for the expressiveness of the function. Equation 1 displays the loss function with trajectory regularization. The current trajectory length is defined as L_i, and the original trajectory length is defined as L_o.

$$Loss = \frac{1}{n}\sum_{n=1}^{\infty}(y_i - \hat{y}_i)^2 + \lambda(\frac{L_i}{L_o}) \tag{1}$$

3.2 Batch Normalization

The authors compare the new trajectory regularization method to a past method called batch normalization. Batch normalization was created after the idea that we always normalize the input data before inputting it through a neural network. If one does not normalize the data beforehand, after multiplying the weights and adding the biases, one could get an input to the activation function similar to the example in Figure 7.

Figure 8: Caption

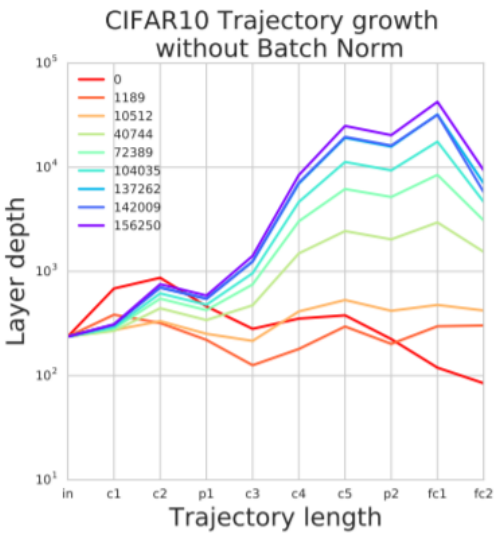


In this figure, the input has a mean of -17, which skews all of the values to be almost zero.

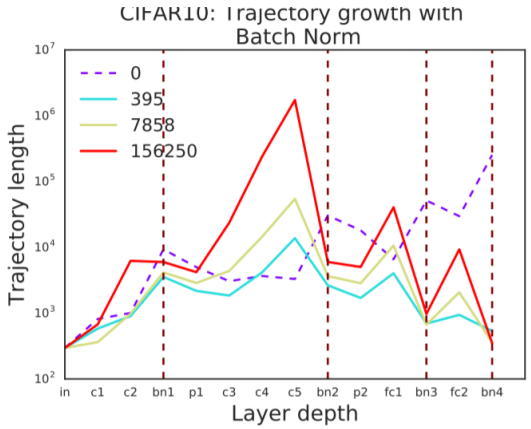
One can use equation 2 as a method to normalize values.

Figure 9: Batch Normalization and Trajectory Length

(a)



(b)



$$x_{new} = \frac{x - \mu}{\sigma} \tag{2}$$

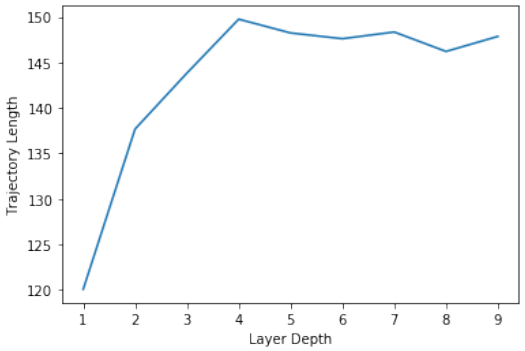
Since we do this for the first layer, why don't we do this for hidden layers? Batch Normalization addresses this issue and does normalization in some of the hidden layers to ensure that we get values that are not skewed on the sigmoid function. Given how batch normalization is a form of regularization, we can expect that neural networks with the method have lower measures of expressivity. The authors actually prove this by creating neural networks without batch normalization and measuring the trajectory length and then later recreating the same neural networks with batch normalization and measuring the trajectory length. The authors prove that batch normalization returns a lower measure of expressivity by displaying the differences in trajectory length as seen in Figures 8 and 9. My results confirm that the authors' new method of measuring expressivity.

I reproduce the authors results by creating my own neural network in Figure 10 and get similar values in Figure 9. We can see that in Figure 10.b that I get much less deviation than in Figure 9.a. Figure 10.a is similar to Figure 9.a in the way that they both increase from layer to layer. Figures 10.b and 9.b are similar in the way that the normalization halts the trajectory length from becoming large.

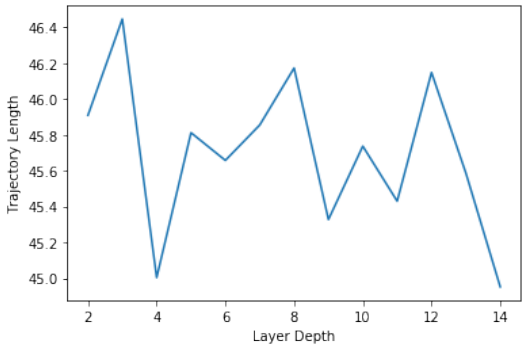
Since there is already a fairly large consensus that batch normalization decreases the chance that a small change in the inputs will create a large change in the outputs [6](which falls under the definition

Figure 10: Reproducing Results of Batch Normalization and Trajectory Length

(a) Reproduced without Batch Norm



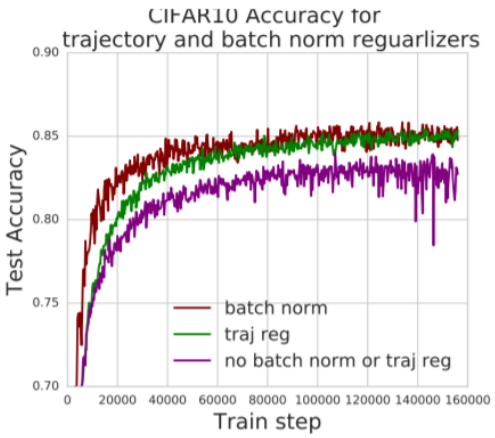
(b) Reproduced with Batch Norm



of neural network expressivity), the results of batch normalization decreasing the trajectory length helps prove that trajectory length is a good measure of expressivity.

In order to see how effective the new trajectory regularization method is, the authors train a set of neural networks with batch normalization, trajectory regularization, and without any form of regularization. In Figure 11, it is fairly clear that trajectory regularization helps increase the accuracy of a neural network over epochs. Figure 11 also shows how batch normalization actually out performs trajectory regularization. One problem with batch normalization is that it requires a lot of computing power, which is why it may not be the best alternative for some cases, when one may need regularization.

Figure 11



4 Discussion

The authors of *On the Expressive Power of Deep Neural Networks* describe new methods to measure the expressivity of neural networks. These new methods allow anyone applying neural networks to a specific field to understand how their outputs will differ with a small change to their inputs.

Neural networks are very popular and also very new to the world of predictive models. Since they are so young, having proper measures to determine the effectiveness of a neural network are still developing and the purpose of *On the Expressive Power of Deep Neural Networks* is to address new measures of expressiveness and in the process create a new method of regularization. After this paper, the authors of created a literature review paper on Neural network expressivity [4], and for the most part just summarize their work, while adding parts of other people's work. It is clear that more work is needed in neural network expressivity, since there are not a lot of papers on the subject. In the future, I believe expressivity should be focused on more because understanding why a neural network outputs certain values is very important.

There has been more of a boom lately with papers and books challenging the black box understanding of neural networks [7, 8] and pushing for a more mathematical understanding. I believe this push is necessary for improving the predictive modeling of a neural network.

References

[1] Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., Sohl-Dickstein, J.: On the expressive power of deep neural networks. arXiv:1606.05336 (2016)

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[3] Zhang G. Peter Times series forecasting using a hybrid ARIMA and neural network model. Neurocomputing 2003;50:159–75.

[4] Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. (2016). Survey of Expressivity in Deep Neural Networks. ArXiv e-prints.

[5] G. HE, Z. CAO, P. ZHU and H. OGURA: Controlling chaos in a chaotic neural network. Neural Networks, 16 (2003), 1195–1200.

[6] Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of The 32nd International Conference on Machine Learning, 448–456.

[7] P. Cortez and M. Embrechts. Opening black box data mining models using sensitivity analysis. In IEEE Symposium Series in Computational Intelligence 2011 (SSCI 2011), Paris, France, 4 2011

[8] Castelvecchi, D. Can we open the black box of AI? Nature 538, 20–23 (2016)