THE USE OF LYAPUNOV EXPONENTS IN DEEP NEURAL NETWORKS

Tirthankar Mittra[1]

INTRODUCTION:

Neural Networks are ubiquitous, models like ChatGPT and BERT [6] have created a storm in many industries but training such big models can take several days with huge computation power, among several problems it contributes to global warming I propose it's better to invest time at the beginning and select hyperparameters like activation function, learning rate, regularization method, etc. and initial points properly. In this work I propose how Lyapunov exponents can be used to help in making these design choices. Although my work focuses on Deep Neural Networks it can be generalized to other Machine Learning techniques like Support Vector Machine (SVM). In this paper I have explored two design questions for a Dense Neural Network in light of Lyapunov exponents.

In my first experiment I used Lyapunov Exponents to select good initial model parameters of a Dense Neural Network (DNN). Initial model weights are generally selected randomly from a probability distribution such as Gaussian. Optimization techniques like gradient descent, newton's method is then used to find local minima of the loss function. In this project I plan to explore how the largest Lyapunov exponents can be used to make better initial guesses.

For the second experiment I explore how Lyapunov exponents can be used to select some hyperparameters of a DNN. Hyperparameters are model parameters which are specified by the user and remain fixed during the training of DNN, a huge chunk of work in machine learning is to find a good set of hyperparameters, there are multiple strategies like using expert knowledge or using validation set to select hyperparameters.

This report is divided into five sections: INTRODUCTION, RELATED WORKS, METHODOLOGY, RESULTS and CONCLUSION. The METHODOLOGY section goes into detail about what experiments I performed and my reasoning for making certain choices whereas the RESULTS section discusses the results of my experiments and their implications.

RELATED WORKS:

To the best of my knowledge, there has been no previous published work on the use of Lyapunov exponents to make good design choices for a Dense Neural Network (DNN). As for my first experiment of choosing good initial set of model parameters there have been different strategies that exist for selecting initial weights, such as random initialization, pre-training [1], initialization based on activation function [2], architecture, and domain-specific initialization [3]. I want to explore a new strategy of selecting initial weights for a generic optimization problem, although my method could be extrapolated to any optimization problem for the purpose of simplicity, I will be focusing on Deep Neural Networks. My methodology is to initialize the model weights for my optimization problem with multiple(N) randomly chosen initial points $\{x_0, x_1..x_N\}$ and then start doing gradient descent, I will also do gradient descent from neighboring points of $\{x_0, x_1..x_N\}$ and calculate the largest Lyapunov exponent (λ_1), similar to Kantz's[4] algorithm. My reasoning for selecting the initial point with the least λ_1 as the best point for the optimization problem is because all nearby trajectories will converge faster for such points to the local minima, but after conducting multiple experiments my initial reasoning has changed which is discussed in detail later in the paper. Although Kantz's algorithm is an improvement over Wolf's algorithm [5] because it takes multiple neighboring points to approximate the biggest Lyapunov exponent, this can be computationally more expensive, especially for Neural Networks which is a very high dimensional problem. In future I plan to replace Kantz's algorithm for calculating Lyapunov exponents with Wolf's because it will give me computation speed.

There have been lots of revolutionary activation functions discovered which have improved the performance of Neural Networks, the ReLU [7] activation function is an example of such an activation function. All activation functions are not suitable for all problems, it's a hyperparameter selected prior to the training of Neural Nets, my second experiment proposes a scheme to determine which activation function is better for a particular optimization problem.

METHODOLOGY:

Lyapunov exponents are mathematical quantities used to describe the behavior of dynamical systems. They provide a measure of how quickly nearby trajectories in the phase space of a system diverge or converge over time. In this research work I calculate the Lyapunov exponents locally for a Dense Neural Networks (DNN) and investigate how it can be used to design a better DNN architecture. In general, optimizing neural networks is not a convex optimization problem. The reason for this is that neural networks typically have multiple local minima, saddle points, which means that the optimization problem does not have a unique global minimum. This non-convexity makes the optimization problem challenging.

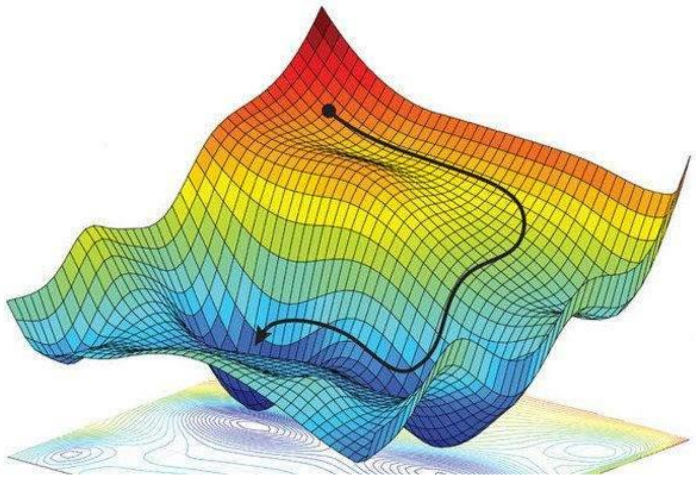
Fig 1. DNN optimization landscape.

I would like to highlight the similarity between the evaluation of the next point in the trajectory of a Lorenz system with given ODEs using Runge-Kutta, and the evaluation of the next point in the optimization landscape of a DNN using SGD (stochastic gradient descent). The gradient indicates the direction of steepest change, which can be likened to moving down the steepest slope of a hill. In SGD you start from an initial point and move down the steepest slope until you arrive at a potentially good local minimum. Fig [1] shows this downhill trajectory traced by an initial point. In short, the state variables (x, y, z) in Lorenz system is equivalent to model weights and bias terms of a DNN, the next point in the trajectory for DNN is found using SGD** whereas for systems with given ODEs Runge-Kutta is used.

****Note SGD is only one of the techniques for optimizing model parameters, there are other methods like Newton's method.**

After establishing this analogy, it becomes easy to calculate the Lyapunov exponents using Kantz algorithm [4]. Creating this analogy is important because it opens many tools and techniques used for analyzing non-linear dynamics. The DNN optimization landscape is influenced by various factors, including loss functions, training inputs and outputs, hyperparameters such as activation function, number of layers, learning rate, regularization strategy etc.

$$\text{MSE} = \frac{1}{N}\sum_{i}^{N}(Yi - \hat{Y}i)^2$$

Eq.1

$$\text{Log loss} = \frac{1}{N}\sum_{i=1}^{N} -(y_i * \log(p_i) + (1-y_i) * \log(1-p_i))$$

Eq. 2

I conducted all my experiments with two loss functions MSE (mean squared error) [Eq. 1] and binary cross entropy [Eq. 2]. This was done to ensure that the trends I noticed remained the same across the two popular types of loss function. I created a pseudo-optimization problem to be solved by my DNN for my experiments. To generate the pseudo training data, I used three binary input variables (x_0, x_1, x_2) and generated the XOR between them, the XOR would serve as my output, I then introduced gaussian noise to my input variables (x_0, x_1, x_2) with zero mean and 0.3 standard deviation.

As can also be seen from Fig [1], starting at different initial conditions in the optimization landscape will take you to a different local minimum, in my first experiment I use local Lyapunov exponents to explore what is a good initial starting point for my optimization problem. My intuition behind this idea is that for a good starting point its neighboring points will converge faster so it'll have a more negative Lyapunov Exponent and hence a lower final loss. Fig [2] show these dynamics, "Point 1" & its neighbor "Point* 1" converge faster in the image on the right, obviously it will have a more negative Lyapunov Exponent and my claim is that this will also correspond to a lesser final loss.

My second experiment is to use Lyapunov exponents to find a good activation function. Fig [3] shows the DNN architecture I used, it has two hidden layers and a final output layer. The activation function in the final layer is sigmoid which is fixed, but for the two hidden layers I changed the activation function to 'sigmoid', 'ReLU' and 'tanh' and calculated the

Lyapunov exponents along with the final loss value. Activation function is essential to the functioning of Neural Nets, it introduces nonlinearity in Neural Networks which otherwise would be a plain high-dimensional linear model not capable of making all the beautiful predictions it makes.

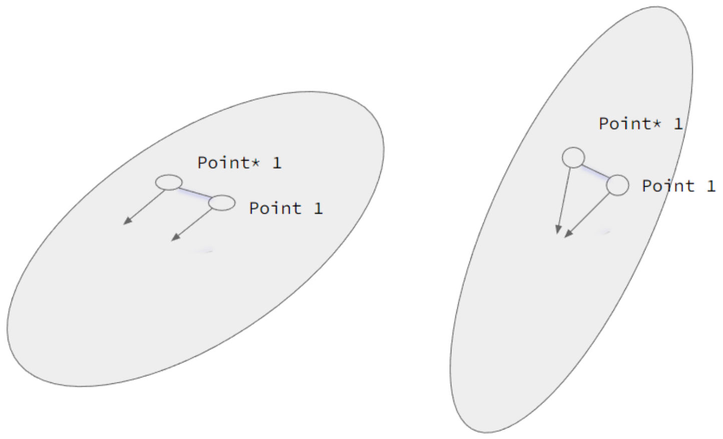

Fig 2. Different types of optimization landscape.

This scheme can be used to select amongst various hyperparameters like learning rate, optimization techniques (SGD, Adam, etc.), regularization, etc. All Lyapunov exponent comparisons are valid between two different DNN models if the number of model parameters are fixed between them. I have not analyzed how the comparison is affected if the number of model parameters change (i.e., more layers or more neurons per layer) between the two models.

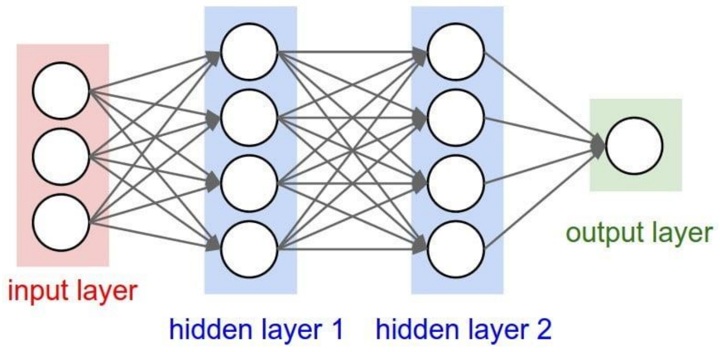

Fig 3. My Dense Neural Network (DNN) architecture.

The algorithm I used is very simple in principle. Take M random points in the observation space, for each of the M points calculate N neighboring points and use Kantz's algorithm to determine the Lyapunov exponents till i_1 number of iterations, then do i_2 number of further iterations and make observations. For a good initial guess of model parameters, I found out that the way Lyapunov exponents relate to the final loss is dependent on how long the Lyapunov exponent was evaluated for. If the Lyapunov exponents were calculated for small duration of iterations, then the good points will have a more negative Lyapunov exponents as they will be converging faster but if the Lyapunov exponents are evaluated for a longer duration of time the neighborhood of good points won't change much as they have already reached a local minima, this is my hypothesis for the observations I made which are discussed in more detail in the results section.

RESULTS:

My expectation as discussed previously was to see a more negative Lyapunov exponents associated with less loss because more negative Lyapunov Exponent means that one is quickly descending down the hill and will reach a local minima faster, what I didn't account for is that the point which is good will reach the local minima faster and once a point reaches near a local minima the neighboring point will not converge quickly with each other as the value of gradients are very less near local minima. This will result in good points having less negative Lyapunov exponents compared to bad initial guesses where the points are still descending the slope. The neighborhood of a good point will converge quickly initially and then it'll will change less, so the relationship between Lyapunov exponents and loss function changes depending on how far one has calculated the Lyapunov exponents. Fig [4] and Fig [5] exactly show this relationship. In Fig [4] the Lyapunov exponent calculation was terminated earlier so one can see a positive correlation between more negative Lyapunov exponents and less final loss, the *p-value* was *0.0796* (as an aside the lesser the value of *p* the more statistical significance the results hold, a *p-value* of 0.5 denotes that it's 50% likely that the null hypothesis is true) with correlation coefficient of *0.2920*. In Fig [5] I calculated the Lyapunov exponents a bit longer, since good initial guesses reach the local minima faster and the neighboring points don't change much the relationship gets flipped, the *p-value* was 0.0317 and the correlation coefficient was -0.2149. If Lyapunov exponents are calculated for a very short duration of time, then it's meaningless but if Lyapunov exponents are calculated a for a longer time getting something like Fig [4] is difficult so it's better to calculate the Lyapunov exponent a little longer and use the negative relationship in Fig [5] to choose a good initial guess.

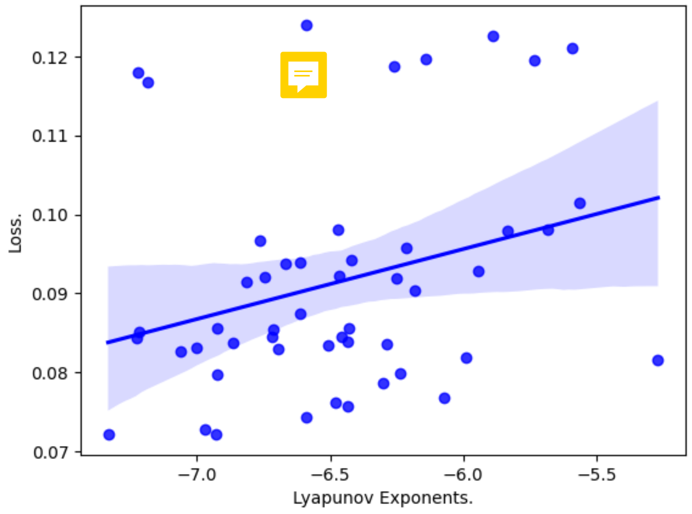

Fig. 4 Lyapunov Exponents calculated for a shorter duration of time.

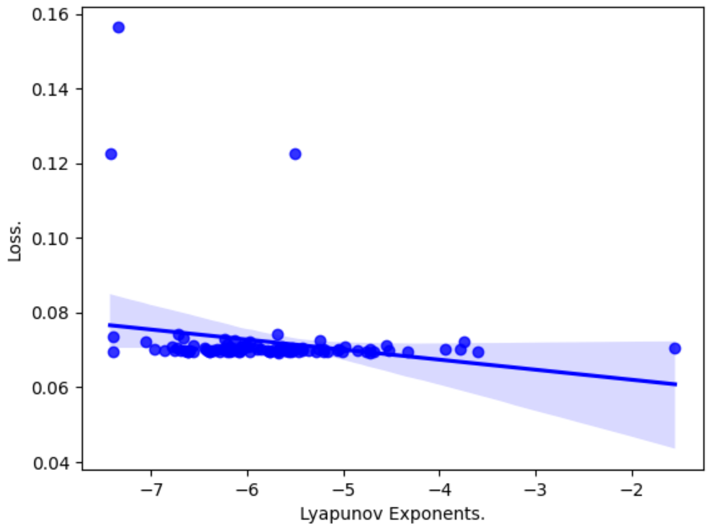

Fig. 5 Lyapunov Exponents calculated for a longer duration of time.

For the second experiment I have three DNN architectures with different activation functions in both the first and the second hidden layers, Fig [3]. Table [1] below highlights the relationship between Lyapunov exponents and the final MSE loss, it can be observed that less negative Lyapunov exponents is associated with lesser final MSE loss, this is because a good activation function takes the initial set of points quickly to a local minimum from where the change in neighboring points with respect to the original point is minimal. I evaluated this result multiple times and the trend remained the same even when the loss function was changed to binary cross-entropy. As discussed in the first experiment when Lyapunov exponents are calculated for shorter time the trend in Table [1] flips but it's also difficult to exploit this reverse trend. From my prior knowledge I know that ReLU is the best activation function for my problem, so it validates the results I got from my experiments.

Activation Function Type	Lyapunov Exponent	Final MSE Loss
Sigmoid	-7.4617	0.24907552301883698
Tanh	-6.5941	0.09893139898777008
ReLU	-6.3162	0.08987971395254135

Table. 1

CONCLUSION:

Clearly there is a relationship between the calculated Lyapunov exponents, and various design choices of a DNN i.e., initial model weights, and activation function. This relationship can be utilized to build a better DNN architecture. In my work I have only shown how one hyperparameter i.e., activation function can be selected using Lyapunov exponents, my conjecture is that this method can be used to select other hyperparameters like regularization and optimization technique, etc. Also, the DNN architecture I used is very small compared to modern standards. I would like to explore how my technique would perform, for example in an AlexNet variant of the ImageNet challenge.

BIBLIOGRAPHY:

[1] Lu, S., Lu, Z., Zhang, Y.D.: Pathological brain detection based on alexnet and transfer learning. Journal of computational science 30 (2019) 41–47

[2] Kumar, S.K.: On weight initialization in deep neural networks. arXiv preprint arXiv:1704.08863 (2017)

[3] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings (2010)249–256

[4] Kantz, H.: A robust method to estimate the maximal lyapunov exponent of a time series. Physics letters A 185(1) (1994) 77–87

[5] Wolf, A., Swift, J.B., Swinney, H.L., Vastano, J.A.: Determining lyapunov exponents from a time series. Physica D: nonlinear phenomena 16(3) (1985) 285–317

[6] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

[7] Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted Boltzmann machines." *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010.