CHAOS
Theory and Applications
in Applied Sciences and Engineering

e-ISSN: 2687-4539
RESEARCH ARTICLE
Vol.7 / No.2 / 2025 / pp.154-165
https://doi.org/10.51537/chaos.1588198

A Chaos-Causality Approach to Principled Pruning of Dense Neural Networks

Rajan Sahu [*,1], **Shivam Chadha** [α,2], **Archana Mathur** [β,3], **Nithin Nagaraj** [§,4] **and Snehanshu Saha** [γ,5]

[*]Dept. of CSIS, Birla Institute of Technology and Sciences Pilani, Pilani, India, [α]Dept. of Mathematics, BITS Pilani KK Birla Goa Campus, Goa, India, [β]Dept. of Artificial Intelligence and Data Science, Nitte Meenakshi Institute of Technology (NMIT), Bangalore, India, [§]Complex Systems Programme, National Institute of Advanced Studies, Bangalore, India, [γ]Dept. of CSIS and APPCAIR, BITS Pilani KK Birla Goa Campus, Goa, India.

ABSTRACT Reducing the size of a neural network (pruning) by removing weights without impacting its performance is an important problem for resource-constrained devices. In the past, pruning was typically accomplished by ranking or penalizing weights based on criteria like magnitude and removing low-ranked weights before retraining the remaining ones. Pruning strategies also involve removing neurons from the network to achieve the desired reduction in network size. We formulate pruning as an optimization problem to minimize misclassifications by selecting specific weights. We have introduced the concept of chaos in learning (Lyapunov Exponents) through weight updates and used causality-based investigations to identify the causal weight connections responsible for misclassification. Two architectures are proposed in the current work - Lyapunov Exponent Granger Causality driven Fully Trained Network (LEGCNet-FT) and Lyapunov Exponent Granger Causality driven Partially Trained Network (LEGCNet-PT). The proposed methodology gauges causality between weight-specific Lyapunov Exponents (LEs) and misclassification, facilitating the identification of weights for pruning in the network. The performance of both the dense and pruned neural networks is evaluated using accuracy, F1 scores, FLOPS, and percentage pruned. It is observed that, using LEGCNet-PT/LEGCNet-FT, a dense over-parameterized network can be pruned without compromising accuracy, F1 score, or other performance metrics. Additionally, the sparse networks are trained with fewer epochs and fewer FLOPs than their dense counterparts across all datasets. Our methods are compared with random and magnitude pruning and observed that the pruned network maintains the original performance while retaining feature explainability. Feature explainability is investigated using SHAP and WeightWatchers. The SHAP values computed for the proposed pruning architecture, as well as for the baselines (random and magnitude), indicate that feature importance is maintained in LEGCNet-PT and LEGCNet-FT when compared to the dense network. WeightWatchers results reveal that the network layers are well-trained.

KEYWORDS
Chaos
Granger causality
Neural networks
Lyapunov exponent
Weight pruning

INTRODUCTION

Designing a neural network architecture is critical to developing neural networks for various Artificial Intelligence (AI) tasks, particularly in deep learning. AI, or Artificial Intelligence, refers to the simulation of human intelligence in machines designed to perform tasks typically requiring human cognition. These tasks include learning, problem-solving, reasoning, and understanding natural language. AI systems use algorithms and data to improve their performance over time, often becoming more efficient with experience.

One of the fundamental challenges in designing neural networks is finding the right balance between model complexity and sample size, which can significantly impact the network's performance. In general, a larger network with more parameters (overparameterized) can potentially learn more complex functions and patterns from the data (Prandi *et al.* 2017). However, larger networks may also be prone to overfitting. On the other hand, smaller networks with fewer parameters may not have enough capacity to learn complex relationships in the data. This can lead to underfitting. Therefore, the challenge in network architecture design is to find the right balance between model complexity and sample size, so that the network can learn to generalize well to new, unseen data. In this context, over-parameterized networks (Zou *et al.* 2018; Mohapatra *et al.* 2022) have become increasingly popular in the deep learning era due to their ability to achieve high expressivity and potentially better generalization performance (Shen *et al.* 2019). The idea is to increase the number of parameters in the network beyond what is strictly necessary to fit the training data and remarkable generalization to test data. Pruning techniques should reduce the number of parameters in a neural network without compromising its accuracy.

Manuscript received: 21 November 2024,
Revised: 24 February 2025,
Accepted: 27 March 2025.

[1]f20190572p@alumni.bits-pilani.ac.in
[2]f20190704g@alumni.bits-pilani.ac.in
[3]archana.mathur@nmit.ac.in (**Corresponding author**)
[4]nithin@nias.res.in
[5]scibase.snehanshu@gmail.com

It is important to ponder why we do not simply train a smaller network from scratch to make training more efficient. The reason is that the architectures obtained after pruning are typically more challenging to train from scratch (Liu *et al.* 2018), and they often result in lower accuracy compared to the original networks. Therefore, while standard pruning techniques can effectively reduce the size and energy consumption of a network, it does not necessarily lead to a more efficient training process.

BACKGROUND AND PROBLEM STATEMENT

Neural network pruning has two main goals to achieve - one being the reduction of model size and the other being a consequence of the former - improved inference efficiency. Traditional pruning methods such as random pruning (Mittal *et al.* 2018b,a) or magnitude-based pruning (Lee *et al.* 2020; Saleem *et al.* 2024) inherently incorporate a form of regularization that could potentially affect or hinder the natural course of learning dynamics of a model. Even though empirically the methods seem to work well, we are interested in a principled approach of pruning that takes the model's task performance into account. Random pruning removes weights in a stochastic manner. Magnitude-based pruning removes weights based on their absolute values, assuming a certain relationship between the magnitude of weights and contribution in the final model performance (Medhat *et al.* 2023). These methods fundamentally don't operate under a static or predefined criterion that does not dynamically adapt to the model's learning trajectory. A key limitation of these approaches is that they impose pruning decisions as an implicit form of regularization rather than integrating pruning dynamically into the optimization process itself.

We pose a broad *Research Question* here: Is there a principled approach to pruning overparameterized, dense neural networks to a reasonably good sparse approximation such that performance is not compromised and explainability is retained? It is well known that dense neural network training and particularly weight updates via SGD have some element of chaos (Zhang *et al.* 2021; Herrmann *et al.* 2022). We expect that, between the successive weight updates due to SGD and miss-classification, there is some observed causality and non-causal weights (parameters) can be pruned, leading to a sparse network i.e. some weight updates cause a reduction in network (training) loss and some do not! Can we train a dense network till a few epochs to derive a pruned architecture for the derivative to run for the remaining epochs and produce performance metrics in the $\epsilon-$ball of the original, dense network? Does this sparse network also train well, verified with Shapley (Lundberg and Lee 2017) and WeightWatcher (*WW*) (Martin *et al.* 2020) tests? Specifically, we contribute to the following:

- Present a unique and unifying framework on chaos and causality for deep network pruning. The unifying framework uses Lyapunov Exponent (LE) (Kondo *et al.* 2021) and Granger causality (GC) (Granger 1969) tandem.
- Propose novel pruning architectures, Lyapunov Exponent Granger Causality driven Fully Trained Network (LEGCNet-FT) and Lyapunov Exponent Granger Causality driven Partially Trained Network (LEGCNet-PT).
- LEGCNet-FT and LEGCNet-PT compare very well in performance and other baselines, Random (Liu *et al.* 2022) and Magnitude based (Li *et al.* 2018) pruning techniques.
- Establish feature consistency of LEGCNet-FT and LEGCNet-PT in explainability.
- Verify empirically that the proposed architectures for pruning are not over-trained and obviously not overparameterized but can still generalize well, on diverse data sets while saving FLOPs (FLoating-point OPeration). We accomplish this via the *WW* test.

We approach pruning as a constrained optimization problem rather than a strict regularization step. Instead of focusing purely on reducing network size, we ensure that pruning decisions adhere to model performance, directly accounting for accuracy and fewer epochs. Using Lyapunov Exponents (LE) and Granger Causality (GC), we identify non-causal weights parameters that do not significantly contribute to loss minimization and selectively remove them. Unlike traditional methods that prune weights based on static criteria, our approach dynamically adapts pruning decisions based on learning dynamics. Our LEGCNet-FT (Fully Trained) and LEGCNet-PT (Partially Trained) frameworks validate this approach. LEGCNet-FT performs pruning after full training, while LEGCNet-PT identifies and removes non-causal weights early in training, significantly reducing computational overhead without compromising accuracy. This adaptive strategy ensures that pruning is an integral part of optimization rather than an afterthought.

The remainder of the paper is organized to present the key methodologies used to develop the pruning technique, followed by a detailed experimental setup and strong empirical evidence of the proposed technique in contrast to the baselines. In summary, we propose pruning techniques, *LEGCNet-FT and LEGCNet-PT* which perform at par with the dense, unpruned architecture and the existing pruning baselines. *In methods like magnitude/random pruning, the choice of percentage pruning for a specific dataset or network is often arbitrary and difficult to select. Our method gives a natural threshold for weight pruning, unlike random, magnitude, and other pruning methods. While maintaining consistent performance, these techniques also help reduce epochs to converge and FLOPs to compute while maintaining feature consistency with their dense counterparts and ensuring proper training across layers validated via WW statistics.*

We claim the following:

- Non-causal weights are identified as the ones which do not impact the accuracy and therefore must be removed from the fully connected network. Such a pruned network performs as well as the dense counterpart, retains the feature explainablity of the dense sibling.
- *Why is feature consistency IN SHAP (explainability test) important for pruned networks?*.
- Such a pruned network satisfies the network health diagnostic test (Weightwatcher (*WW*)) and also incurs reduced floating point operations (FLOPs). Additionally, the stable equilibrium in the loss landscape can be obtained if we can construct a suitable Lyapunov (Energy) function. The motivation for pruning is strongly tied to the health diagnostic tests for DNNs, *WW* in particular. This is because it is possible to establish all layers of a DNN to be correctly trained and therefore not contribute to the generalization gap. This indicates some induced, implicit self-regularization via pruning. Therefore, favorable *WW* Statistics is a strong indicator of the pruned network's spectral distribution being heavy-tailed i.e. the pruned network is correctly trained. This establishes the validity of the pruning mechanism proposed in the paper.

The technical motivation behind our claims is sourced from a diverse set of questions- Why chaos? Does Chaos helps learning and can we support it with experimental results? Do we expect causality between 2 time series sequences weight and loss series? Does it

satisfy the lottery ticket hypothesis by deriving a pruned network by deleting no-causal weights? Wiener-Granger causality is based on the principle that "predictability implies causation". We believe that causality, as estimated by Granger causality, is in any case not to be treated as actual causality, but only an indication of predictability of one time series with the incorporation of information from another time series. This type of predictability also means a kind of redundancy of information and hence informs and justifies the pruning to be carried out. With this view of pruning (of redundant connections) in mind, we need only a reasonable measure of estimating such redundancies in the connections. For this purpose, Granger causality is sufficient. More sophisticated methods such as Transfer Entropy, Compression-Complexity Causality could be tried out in future versions of this work.

It should be noted that the mere use of statistical measures such as correlation or mutual information does not suffice as they lack the directionality of information flow, which is provided by causality measures such as Granger causality and others. The direction of the flow of information is important since we are interested in knowing which connections influence misclassification errors and which ones do not so that pruning can be done appropriately. As per Judea Pearl's ladder of causation, associations (correlations) are at the first rung of the ladder and measures such as Granger causality are at a higher rung. This is a unified view of statistical correlations and causation - seen to be at different levels of measures of information flow (or influence). Chaos, causality, and the manifestation of the Lottery Ticket Hypothesis are the key motivations behind our proposed pruning mechanism.

Chaos and Causality

One way to address the issue of explainability in AI/machine learning is to seek causal explanations for choices made in the learning process. Conversely, a learning process that incorporates choices made out of causal considerations is easier to explain and interpret. This is the motivation behind using causality-based criteria for the choice of what to prune (or not prune) in this study. To this end, we employ Granger Causality (GC) (Granger 1969), one of the earliest and nearly model-free causality testing between two processes (or measurements/time series).

The principle of pruning that is causally informed is formulated as follows. Those connections (weights) in the learning network that do not causally impact the loss are chosen for pruning. To determine the causal impact of a particular connection to the loss, we perform GC between the windowed LE of the weight time series for that connection and the classification accuracy. The rationale behind this is the intuition that the chaotic signature of weight updates supports learning. Biological inspiration for chaotic signatures as a marker for learning is the empirical fact that neurons in the human brain exhibit chaos (Faure and Korn 2001; Korn and Faure 2003) at all spatiotemporal scales. Starting from single neurons to coupled neurons to a network of neurons to different areas of the brain, chaos has been found to be ubiquitous to the brain (Korn and Faure 2003).

Chaotic systems are known to exhibit a wide range of patterns (periodic, quasi-periodic, and non-periodic behaviors), are very robust to noise, and enable efficient information transmission (Nagaraj and Vaidya 2009), processing/computation (Ditto and Sinha 2015), (Kuo 2005) and classification (Balakrishnan *et al.* 2019). There is also some evidence to suggest that weak chaos is likely to aid learning (Sprott 2013). Thus our choice of testing causal strength between LE (a value > 0 is a marker of chaos) and classification accuracy as a criterion for pruning to yield sparse subnetworks that capture the learning of the task at hand.

Gradient Descent and Low Dimensional Chaos

Is the process of updating weights in backpropagation via Gradient Descent chaotic? Is there an alternative interpretation of the minima in the weight landscape via low-dimensional chaos? The weight update in SGD is written as $w_{i+1} \leftarrow w_i - \eta_i \nabla_w f(w_i)$ may be thought of as a discretization to the first order ODE: $w'(t) = -\nabla_w f(w_i)$. The minimizer of the SGD is therefore conceived as a stable equilibrium of the ODE. That is, the minimum, w^* can be thought of as a fixed point to the iterates $w_{i+1} = G(\eta_i, w_i) \equiv w^* = G(\eta_i, w^*)$.

Empirical evidence of chaos in backpropagation: We performed a series of experiments on different datasets to check Sensitive Dependence On Initial Conditions (SDIC) for weight initialization, on a single hidden layer neural network. The weight initialization matrix W_{ij} followed Gaussian distribution $(W_{ij} \sim N(0, \sigma^2))$. We recorded two sets of executions - one with initial weight $w_{11} : w_{ij} \forall i \in 1..h, \forall j \in 1..n$, where n, h are the number of input and hidden neurons - and another, with infinitesimal perturbation $(w_{11} + \delta)$ keeping other parameters same. Each time, the network was trained via gradient descent, and weight series were recorded. The method was repeated for the second weight connection $w_{ij}, i = 1, j = 2$. Later, the Lyapunov Exponents (LE) were computed using the TISEAN package (Hegger *et al.* 1998) on the recorded weight series to measure the perturbed trajectory due to initial perturbation δ. We observed positive LE which marked the presence of some chaotic behavior in gradient descent.

In other words, to introduce perturbation in our experiments, we initialize the weights of the dense neural network and run the experiment under normal conditions. Once a baseline is established, we induce a controlled perturbation on a single weight connection. Specifically for a given weight, we introduce a small perturbation modifying it to the original weight, added with an infinitesimally small value before we start training the model. For example, a weight w0 is initialized to w0=2.0 in the first run, and after adding a small perturbation, w0=2.0001, we rerun the backpropagation capturing the weight updates each time. The purpose of this perturbation is to analyse the sensitivity of the model to slight alterations and observe how these changes propagate through the learning process. We try to investigate if the learning process is chaotic. The computation of Lyapunov exponents and their positive values confirms the presence of weak chaos during backpropagation.

Lottery ticket hypothesis

The "lottery ticket hypothesis" (Frankle and Carbin 2018) is a concept in neural network pruning that suggests that within a dense and over-parameterized neural network, there exist sparse subnetworks that can be trained to perform just as well as the original dense network. Any fully connected feed-forward network $f^d(x; \phi)$, with initial parameters ϕ when trained on a training set D, f^d achieves a test accuracy a and error e at iteration j. Our work LEGCNet, validates the lottery ticket hypothesis by finding the "winning ticket", m, to construct the sparse network, f^s such that $acc^s \geq a$ and $j^s < j$ where $\|\phi\| \gg \|m\|$.

MATERIALS AND METHODS

Dense Neural network - Let f^d be a dense neural network of depth l and width h defined as

$$f^d(x) = W_i^d \sigma_i(W_{i-1}^d \sigma_i(...W_1^d(x))) \tag{1}$$

where W_i^d is the weight matrix for layer i such that $i \in 1..l$.
Sparse Neural Network - Let f^s be sparse neural network of the same architecture as f^d, with depth l and width h.
Two approaches: LEGCNet-FT and LEGCNet-PT - To validate the working of LEGCNet, we divided the method into two discrete approaches. In one approach, the entire training weight series is used for computing LEs, testing GC, and for the identification of causal weights. Essentially, the dense network is trained till convergence and the approach is called LEGCNet-Full Train (LEGCNet-FT). In the second approach, LEGCNet-PT, the network is trained only till certain epochs (10% of the total iterates) and these few weight updates are used for identifying the causal weights.

Granger causality as a tool

We have used popular model-free/ data-driven methods such as Wiener-Granger Causality or G-causality. Granger causality or G-causality works on the principle of modeling the two processes X and Y as auto-regressive processes. Specifically, to determine if '*Y G-causes X*', the two models considered are:

$$X(t) = \sum_{\tau=1}^{\infty} (p_\tau X(t-\tau)) + \sum_{\tau=1}^{\infty} (r_\tau Y(t-\tau)) + \varepsilon_c, \tag{2}$$

$$X(t) = \sum_{\tau=1}^{\infty} (q_\tau X(t-\tau)) + \varepsilon, \tag{3}$$

where t stands for time, p_τ, q_τ, r_τ are coefficients at a time lag of τ and $\varepsilon_c, \varepsilon$ are error terms. Covariance stationarity is assumed for both X and Y. Whether Y G-causes X (or not) can be predicted by the measure known as F-statistic which is the log ratio of the prediction error variances:

$$F_{Y \to X} = \ln \frac{var(\varepsilon)}{var(\varepsilon_c)}. \tag{4}$$

If the model represented by equation (2) is a better model for $X(t)$ than equation (3), then $\text{var}(\varepsilon_c) < \text{var}(\varepsilon)$ and $F_{Y \to X} > 0$, suggesting that *Y Granger causes X*. Even though G-causality uses the notion of autoregressive models for the variables, the generic nature of this modeling with minimal assumptions about the underlying mechanisms makes it a popular choice in a wide range of disciplines.

Lyapunov exponents as a tool

In dynamical systems, Lyapunov exponents are used to measure the rate of divergence of infinitesimally close trajectories, and they are an important tool for characterizing the behavior of chaotic systems. In a chaotic system, even small differences in initial conditions can lead to large differences in the behavior of the system over time. Lyapunov exponents provide a way to quantify sensitivity to initial conditions, and they can be used to predict the long-term behavior of a system. Systems with positive Lyapunov exponents are considered to be chaotic, while systems with negative Lyapunov exponents are stable and predictable. The magnitude of the Lyapunov exponents provides information about the rate of divergence or convergence of close trajectories, and this can be used to study the dynamics of neural network training (via gradient descent) in a quantitative way. It's worth noting that the Lyapunov exponent is a sensitive measure of chaos and requires careful numerical computation.
A differential equation defining a continuous-time smooth dynamical system in n dimensions is given as $\dot{x} = f(x)$ where $f : U- > R^n$ is a continuous function and $x(t) \in R^n$ is a state variable at time t. A map $f : U- > R^n$ defines a discrete-time smooth dynamical system in n-dimensions, as $x_{t+1} = f(x_t)$ where x_t is the state of the system at time t. Let there be two points x_0 and $x_0 + \delta_0$ which are separated by a small vector δ at t=0. At time t, the rate of separation of the two neighboring points, as they travel in a chaotic region, is given as $|\delta_t| \approx |\delta_0| e^{\lambda t}$ where λ is the LE, δ_0 is the difference between the two trajectories at t=0 and δ_t is the difference at t=t. The λ is calculated as

$$\lambda_{f(x)} = \lim_{\delta_0 \to 0} \lim_{n \to \infty} \left\{ \frac{1}{n} ln \frac{|\delta_t|}{|\delta_0|} \right\} \tag{5}$$

Different methods exist for calculating the Lyapunov exponent, such as the Wolf algorithm (Wolf *et al.* 1985) or the Rosenstein method (Rosenstein *et al.* 1993), which have specific considerations depending on the system being studied.

Windowed Weight Updates

Consider a neural network with n inputs and r outputs, p hidden layers of h neurons, and, the input vector denoted as $x \in R^n$, The network when trained by SGD generates a sequence of weight updates represented by $w^{ji} = \left[w_0^{ji}, w_1^{ji}, ..., w_s^{ji}, ..., w_S^{ji} \right]$ where w_s^{ji} is the weight of the ith neuron of the input layer and the jth neuron of the hidden layer at the sth iteration. Considering, the weights being collected for the initial few epochs, the weight iterates for the hidden layer and output layer are $W_h = \left\{ w^{ji} \right\}, W_o = \left\{ w^{kj} \right\} \quad \forall i \in \{1,..,n\}, \forall j \in \{1,..,m\}, \forall k \in \{1,..,r\}$ An infinitesimal perturbation δ_0 is introduced in the initial weight w^{11}, given as $w^{11\delta_0} = w^{11} + \delta_0$, keeping other parameters - weights (initialization), learning rate, optimizer, and loss function- same. The network is then retrained with the perturbed weight, and the weight updates are recorded again as $W_h^{\delta_0} = \left\{ w^{ji\delta_0} \right\}, W_o^{\delta_0} = \left\{ w^{kj\delta_0} \right\} \quad \forall i \in \{1,..,n\}, \forall j \in \{1,..,m\}, \forall k \in \{1,..,r\}$ A difference series obtained by subtracting perturbed weights from initial weights is $\delta W_h = \left\{ \delta w^{ji} \right\}, \delta W_o = \left\{ \delta w^{kj} \right\} \quad \forall i \in \{1,..,n\}, \forall j \in \{1,..,m\}, \forall k \in \{1,..,r\}$. We divide the weight series δw^{ji} into D windows, $w^{ji} = \bigcup_{l=1}^{D} w^{ji(l)}$, and compute the LEs of all the windowed-weight trajectories. The series of the LE λ of the windowed-weight trajectories $w^{ji(D)}$ are represented using the notation $\left\{ \lambda^{ji\{1\}}, \lambda^{ji\{2\}}, ..., \lambda^{ji\{D\}} \right\}$. Additionally, we record the accuracy at every window during training $\forall l \in 1, \ldots, D$, captured for weights at $w^{ji(l)}$ and $w^{kj(l)}, \forall i \in \{1,..,n\}, \forall j \in \{1,..,m\}, \forall k \in \{1,..,r\}$. After computing the series of windowed-Lyapunov exponents and corresponding *wls*, we test the Granger Causality (GC) between them. The F-statistics of the two series are computed, and the p-value is measured to check whether the windowed-Lyapunov exponents "Granger-caused" the loss, *wls*. If so, then this reveals that certain weight connections are causing the loss, and they are pruned from the network. Hence, the weights that resulted in Lyapunov exponents "Granger-causing" the loss, are chosen for pruning. Before checking the GC, we confirmed that the series is stationary.

Approximation capability of LEGCNet

The approximation capability of LEGCNet is explored here, that a sparse network, f^s, and dense network, f^d are ϵ close to each other if the activation function used in the network satisfies the Lipschitz property . We have theoretically established that the AF sigmoid is Lipschitz (Lemma 1) and showed the sparse network LEGCNet, approximates the dense counterpart.

The motivation of the theoretical analysis is taken from (Qian and Klabjan 2021) which states that the approximation abilities of dense and sparse networks are ϵ close only if an activation function is L1 Lipschitz. The main theorem and lemma in this section build that the LEGCNet pruned network (f^s) is ϵ-close to f^d with probability 1-δ (Qian and Klabjan 2021). Consider f^d to be a dense neural network in Figure 1 defined as

$$f^d(x) = W_i^d \sigma_i(W_{i-1}^d \sigma_i(...W_1^d(x))) \tag{6}$$

where W_i^d is the weight matrix for layer i such that $i \in 1..l$ and $h_i > h_0, h_i > h_l, \forall i \in 1...(l-1)$. We assume that for network in (6), σ_i is L_i-Lipschitz and weight matrix W_i^d is initialized from uniform distribution $U[\frac{-K}{\sqrt{max(h_i,h_{i-1})}}, \frac{K}{\sqrt{max(h_i,h_{i-1})}}]$, for some constant K.

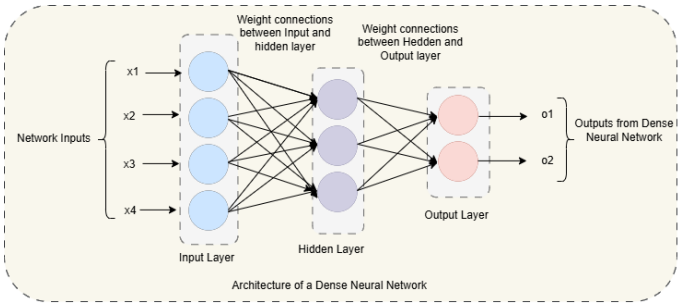

Figure 1 Dense Neural Network

Theorem 0.1. *(Approximation Capability of the Sparse Network) Let* $\epsilon > 0$, $\delta > 0$, $\alpha \in (0,1)$ *such that the, for some constants* K_1, K_2, K_3, K_4, K_5,

$$h \geqslant max\left\{K_1^{\frac{1}{\alpha}}, \left(\frac{K_2}{\epsilon}\right)^{\frac{1}{\alpha}}, \left(\frac{K_3}{\delta}\right)^{\frac{1}{\alpha}}, K_4 + K_5 log\left(\frac{1}{\delta}\right)\right\}$$

then sparse network f^s *obtained from LEGCNet by the mask m, and pruning the weights* $W_i^d, \forall i \in 1...l$ *is* ϵ*-close to* f^d*, with the probability* $(1-\delta)$*, i.e.*

$$\sup_{x \in B_{d0}} \left\| f^s(x) - f^d(x) \right\|_2 \leq \epsilon$$

Remark: Lipschitz property of the activation functions (Saha *et al.* 2020) is a necessary condition to validate the approximation capability of the proposed sparse network. We have used *sigmoid activation* in the sparsely trained/pruned network.

Lemma 0.2. *Sigmoid activation is Lipschitz.*

Proof: If a function $f(x)$ is Lipschitz continuous, then: $\|f(x) - f(y)\| \leq K\|x-y\| \equiv \|f'(x)\| \leq K$. If $K < 1, f$ is a contraction map as well. We know that Sigmoid, $\sigma(x) = \frac{1}{1+e^{-x}}$; and $\|\sigma'\| = \|\sigma(x)(1-\sigma(x))\|$. It's easy to follow that : $\|\sigma(x)(1-\sigma(x))\| \leq \|\sigma(x)\| \|1-\sigma(x)\| \leq C_1, C_2$. Since $0 \leq C_1, C_2 \leq 1$, $C_1 * C_2 = \delta \leq 1$. Hence sigmoid is Lipschitz.

Methodology Overview

Our methodology in Figure 2 gauges causality between weight-specific Lyapunov Exponents (LEs) and misclassification, facilitating the identification of weights for pruning and retention in the network. In the initial step, we train an MLP and record the weights at the end of each iteration (single forward and backward pass), resulting in a time series of weights for the entire training batch. In the first round of training, the weight series is saved for all the weight connections. We introduce a small perturbation in one weight connection and train the network for the second time. Inducing a perturbation is done to discover the chaotic behavior of a NN by investigating the SDIC of the network (the initial conditions - initial weights). After the second round of training, a difference series is computed for all the weight connections.

This process gives insight into changes observed in weight trajectories after introducing a small perturbation to one weight connection. This step captures the impact of the perturbation on the entire network's weights. Once the difference series is obtained, we divide the series into K partitions to obtain the LEs on each partition for every connection. The LEs are then investigated further by computing their causality with misclassification. If, for a particular connection, the causality is established, we infer that the connection hinders learning. The process is repeated for all connections. The causal ones are saved and the non-causal ones become candidates for pruning.

Our pruning method was developed as follows. Initially, we trained a simple Multi-layer Perceptron (MLP) on a given dataset. Throughout the training process, we recorded the weights at each iteration, resulting in a time series of weight values. These weight time series were subsequently utilized to estimate the LE, a measure of chaotic behavior, using the TISEAN package in conjunction with MATLAB scripts. This estimation was performed using a sliding window approach, generating a time series of LEs.

By combining the time series of Lyapunov exponents and the model accuracies obtained at the end of each window, we employed a Granger causality module to investigate whether the weights had a causal relationship with the model's accuracy. This analysis determined whether specific weights "Granger caused" the accuracy. Based on our experiments, the time series of LEs consistently exhibited positive values, indicating the presence of chaotic elements in the weight time series. Consequently, our study focused on understanding whether these weights Granger caused the model's accuracy. Any weights that did not demonstrate this causal relationship were pruned before conducting subsequent model runs (code will be shared on request).

These findings and our pruning methodology contribute to a better understanding of the relationship between weights and model accuracy, enhancing the efficiency and performance of future training iterations. The complete implementation details and code will be shared on request.

EXPERIMENTAL SETUP

In our study, we employed Python3.10 and Matlab R2022a to conduct experiments on a single hidden layer neural network on various datasets. Our experiments were conducted on a Ryzen 9 3900XT Desktop Processor with 32GB RAM and 1TB HDD. During training, we stored the weight updates for every connection in CSV files. We assumed a window size of 200 iterates and computed the LE for each weight connection on every window. Further, we calculated the training and test accuracy on every window, to capture the misclassification rates. Thus, we obtained a sequence of windowed LEs and windowed accuracies for every connection.

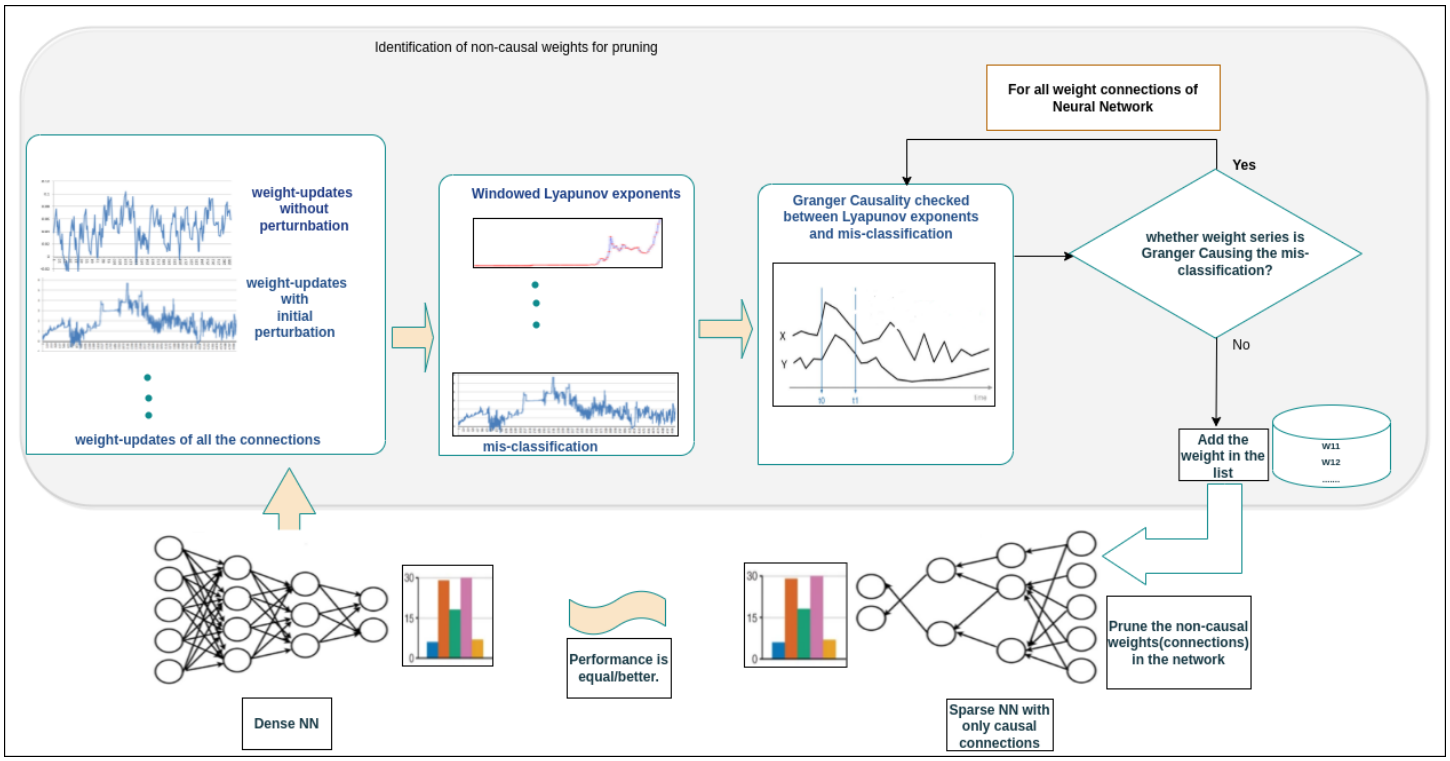

Figure 2 Weights pruned via LEGCNet method for selecting connections in sparse neural network

We then computed the GC between the windowed LEs and the misclassification rate to identify weight connections that Granger caused misclassification.

In the process of pruning the network, we removed the connections for which the LEs were found to Granger cause misclassification. After pruning, we reran the experiment by keeping the same initial weights, optimizers, and other hyperparameters as the unpruned network. We extended the work on seven different datasets and recorded the epochs and accuracies of the pruned network. Interestingly, we observed that the accuracies of the sparse network exceeded those of the dense network. We have used seven datasets - Cancer, Titanic, Banknote, Iris, Iris (3 features), Vowel and MNIST. The datasets we used for experiments are tabular datasets stored as CSV files.

The file contains features as well as class labels associated with the problem(classification) at hand. The features are fed as input to the neural network and the output labels are provided as classes or categories of each corresponding dataset. The code base of a neural network fetches the input and output explicitly. The information about the inputs/outputs (features/labels) of the dense neural network is related to the precise dataset in use and this is publicly available on the UCI Machine learning repository.

In our study, we conducted experiments in two parts. In the first part, we trained the neural network until convergence (LEGCNet-FT) in Figure 3 and computed windowed-LEs for each weight connection as well as windowed accuracies. We then computed the GC and pruned the network by removing connections that were found to cause misclassification. In the second part of the experiment, we trained the network only for a few epochs (LEGCNet-PT) in Figure 4 and used these initial weight updates to compute windowed LEs and accuracies, repeating the same procedure as in the first part of the experiment. Finally, we compared the performance of the pruned network (LEGCNet-FT and LEGCNet-PT) to that of the original network. The results of all these experiments were recorded and presented in tables. The code will be shared on request.

RESULTS AND DISCUSSION

We ran the experiments on seven tabular datasets - Cancer, Titanic, Banknote, Iris, Iris (3 features) Vowel and MNIST. The datasets were divided into 80:20 train-test split and the code was run five times, each maintaining different network initialization. The best results from every initialization are reported in Tables 1 and 2. Table 1 is the comparison of the performance of Dense and LEGCNet-FT. We present the description of table columns.

FLOPs - DN are the Flops needed for training the Dense network, FLOPs - LEGCNet-FT - FLOPs consumed for training the LEGNet-FT network, Non causal Weights - those weights which were found non-causal in Dense Network; Epochs are the epochs needed for Dense and LEGCNet-FT, Accuracy and F1 scores are shown for Dense and LEGCNet-FT. The column percentage pruned (fraction of parameters removed *100) shows the weight connections removed from the dense network.

We compared the FLOPs, % pruned (fraction of parameters removed *100), accuracy, f1-scores, and epochs for all methods (dense, LEGCNet-FT and LEGCNet-PT). Table 1 shows the performance comparison of dense network and LEGCNet-FT. Remarkably, LEGCNet-FT achieves notable reductions in FLOPs without compromising accuracy. Furthermore, LEGCNet-FT converges significantly faster consuming a few epochs compared to the dense network. Specifically, for the Titanic, Vowel, and Cancer datasets, LEGCNet-FT achieves convergence in just half the number of epochs required by the dense network. Nonetheless, both network achieves a similar level of performance without significant differences, thus validating the lottery ticket hypothesis. Table 2 demonstrates the performance of LEGCNet-PT. It shows that

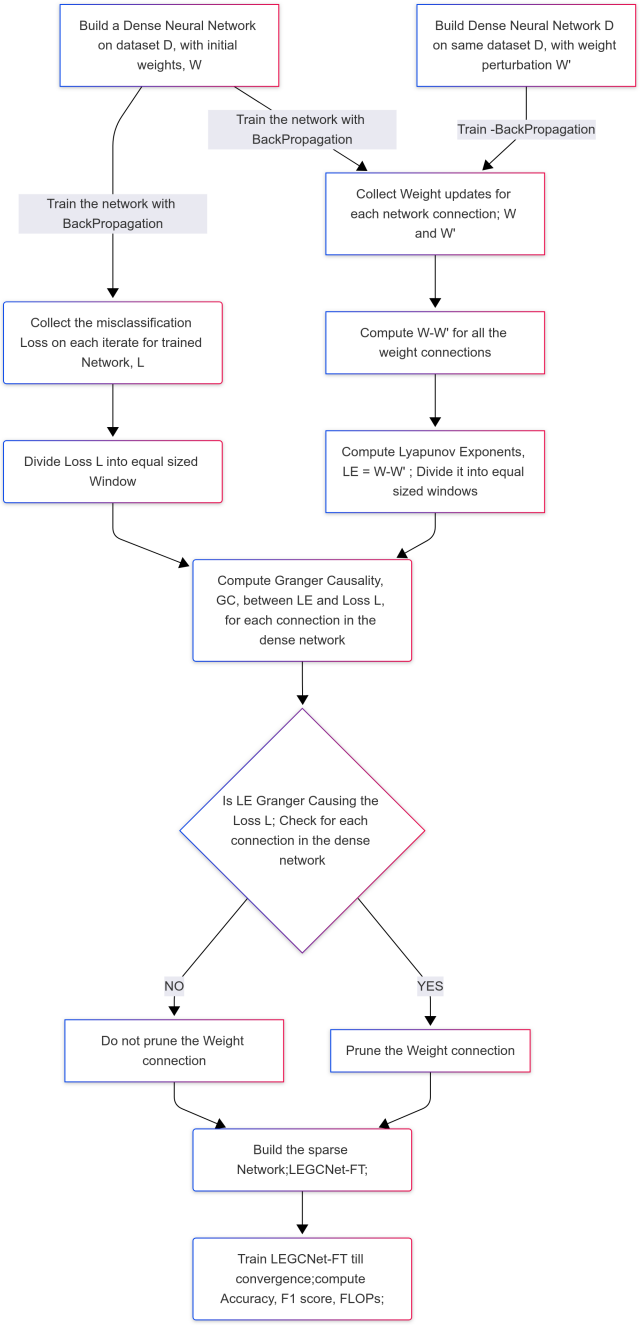

Figure 3 Flowchart for LEGCNet-FT

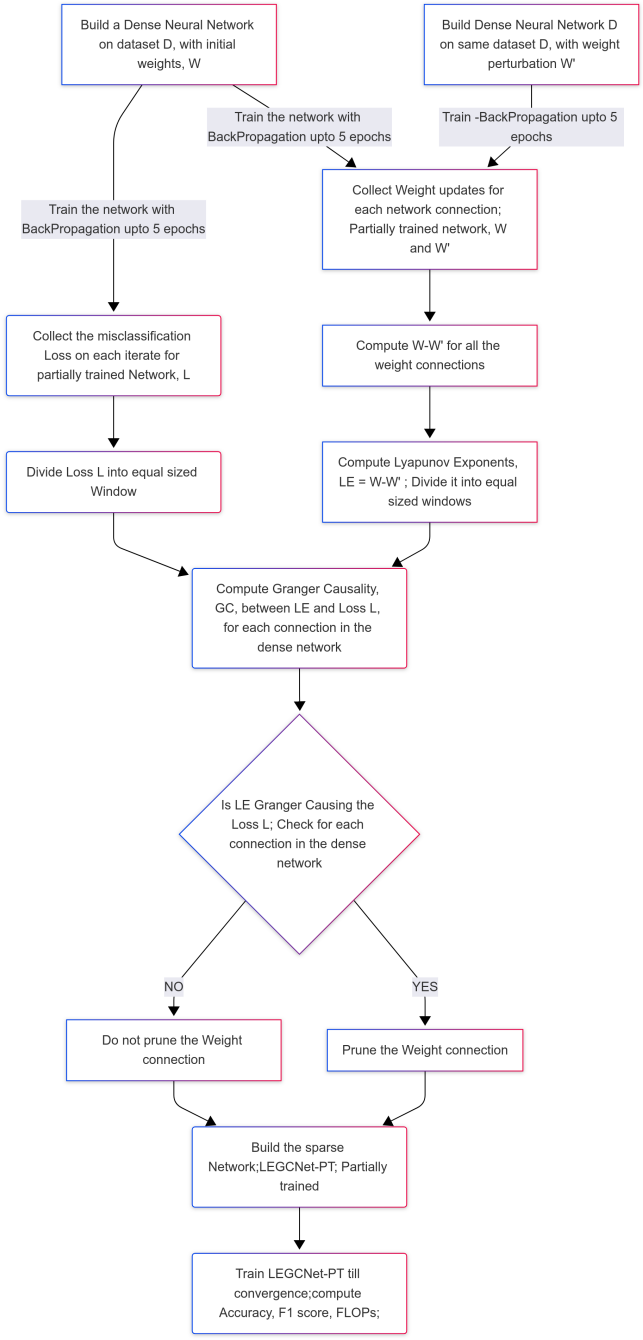

Figure 4 Flowchart for LEGCNet-PT

LEGCNet-PT performs at par with its counterpart, in terms of convergence speed, FLOPs, accuracy, and F1 scores.

We utilized two diagnostic tools, *namely WW and SHAP*. The *WW* validates the compliance of our architecture, as reflected in the Table 3 and Figure 5 plotted for MNIST. *WW* is a powerful open-source diagnostic tool designed for analyzing Deep Neural Networks (DNN). without requiring access to training or test data. It leverages cutting-edge research on the underlying principles of deep learning, specifically the Theory of Heavy-Tailed Self-Regularization (HT-SR).

Drawing inspiration from various fields such as Random Matrix Theory (RMT), Statistical Mechanics, and Strongly Correlated Systems, WeightWatcher provides valuable insights into the inner workings of DNNs and sheds light on why deep learning proves to be effective.

WW analyzes each layer by plotting the empirical spectral distribution (ESD), which represents the histogram of eigenvalues from the layer's correlation matrix. Additionally, it fits the tail of the ESD to a (truncated) power law and presents these fitted distributions on separate axes. This visualization approach provides a clear representation of the eigenvalue distribution and highlights the presence of heavy-tailed behavior in the network's layers. In general, the ESDs observed in the best layers of high-performing DNNs can often be effectively modeled using a Power Law (PL) function. The PL exponents, denoted as alpha, tend to be closer to 2.0 in these cases indicating a heavy-tailed behavior in the layer's

Table 1 Comparison of Dense Networks (DN) and Sparse Networks using LEGCNet-Fully Trained (FT) across datasets. Accuracies and F1-scores are reported on the test set.

Dataset (hidden)	FLOPs DN	FLOPs FT	Non-causal	Epochs DN	Epochs FT	Acc. DN	Acc. FT	F1 DN	F1 FT	% Pruned
Cancer (6)	60	54	6	70	38	0.8759	0.8686	0.8656	0.8570	10.00
Titanic (8)	56	52	4	43	20	0.7225	0.7177	0.6948	0.6843	7.14
Banknote (8)	40	36	4	9	7	0.9018	0.8909	0.9016	0.8906	10.00
Iris (6)	42	40	2	182	166	0.9000	0.9000	0.9124	0.9419	4.76
Iris (3f) (6)	36	26	10	139	126	0.9000	0.9000	0.9330	0.9040	27.78
Vowel (4)	36	34	2	36	14	0.7462	0.7538	0.7307	0.7400	5.56
MNIST (50, 30)	41000	40861	139	27	30	0.9121	0.9165	0.8669	0.8781	0.34

Table 2 Comparison of Sparse Networks where causality is derived from two training strategies: full training (LEGCNet-FT) and partial training (LEGCNet-PT). Iris 3f refers to the Iris dataset with 3 features. *Epochs** indicates the number of epochs used to compute non-causal weights in LEGCNet-PT.

Dataset (*Epochs**)	FLOPs FT	FLOPs PT	Epochs FT	Epochs PT	Acc. FT	Acc. PT	F1 FT	F1 PT	% Pruned
Cancer (12)	54	42	38	24	0.8686	0.8759	0.8570	0.8643	30.00
Titanic (3)	52	29	20	36	0.7177	0.7249	0.6843	0.6969	48.21
Banknote (2)	36	37	7	6	0.8909	0.8945	0.8906	0.8943	7.50
Iris (20)	40	40	166	173	0.9000	0.9000	0.9419	0.9124	4.76
Iris 3f (20)	26	28	126	135	0.9000	0.9333	0.9040	0.9330	22.22
Vowel (5)	34	28	14	20	0.7538	0.7538	0.7400	0.7441	22.22
MNIST (5)	40861	40867	30	29	0.9165	0.9152	0.8781	0.8408	0.32

Table 3 Comparison of Random and Magnitude-based pruning strategies on sparse networks (SN).

Dataset	Epochs	Acc. Rand	Acc. Mag	F1 Rand	F1 Mag
Cancer	42	0.8759	0.8905	0.8656	0.8814
Titanic	35	0.7201	0.7201	0.6842	0.6906
Banknote	6	0.8945	0.9018	0.8943	0.9015
Iris	179	0.9000	0.9000	0.9124	0.9124
Iris (3f)	160	0.9000	0.9333	0.9330	0.9330
Vowel	28	0.7462	0.7462	0.7229	0.7307
MNIST	31	0.9119	0.9121	0.8627	0.8669

Table 4 WeightWatcher summary for models trained with the same initialization on MNIST.

Model	Layer 1 (784–50)		Layer 2 (50–30)		Layer 3 (30–10)	
	α	α_w	α	α_w	α	α_w
Dense	2.19	1.63	1.51	1.88	1.94	2.76
LEGCNet-FT	2.24	1.64	1.51	1.83	2.29	3.20
LEGCNet-PT	2.19	1.71	1.51	1.85	2.73	3.87
Random	2.30	1.53	1.70	1.95	2.00	2.84
Magnitude	2.19	1.63	1.55	1.85	1.96	2.78

correlation matrix. *WW* offers several layer metrics to assess the distinction between well-trained and well-correlated layers from the Marchenko-Pastur (MP) random bulk distribution. By analyzing these metrics, *WW* provides insights into the randomness and heavy-tailed nature of layer distributions. The alpha lies between 2.0 and 6.0 on every layer (Table 4). The ESD plots of the three types of training (dense, LEGCNet-PT, LEGCNet-FT) manifest a heavy-tailed distribution of eigenvalues on each layer indicating the layers are well-trained (Figure 5). A careful observation at Figure 5 reveals the following: ESD plot of a layer, where the orange spike on the far right is the tell-tale clue; it's called a Correlation Trap (LeCun *et al.* 1990). A Correlation Trap refers to a situation where the empirical spectral distributions (ESDs) of the actual (green) and random (red) distributions appear remarkably similar, except for a small correlation shelf located just to the right of 0. In the random ESD (red), the largest eigenvalue (orange) is noticeably positioned further to the right and is separated from the majority of the ESD's bulk. This phenomenon indicates the presence of strong correlations in the layer, which can potentially affect the overall behavior and performance of the network. Figure 5 is the case of well-trained layers. Layers have an overlap of random and original ones when they have not been trained properly because they look almost random, with only a little bit of information present. And the information the layer learned may even be spurious. This is the case of a well-trained layer.

SHAP (SHapley Additive exPlanations) is a game-theoretic technique utilized to provide explanations for the output of machine learning models. By connecting optimal credit allocation with local explanations, SHAP employs the well-established Shapley

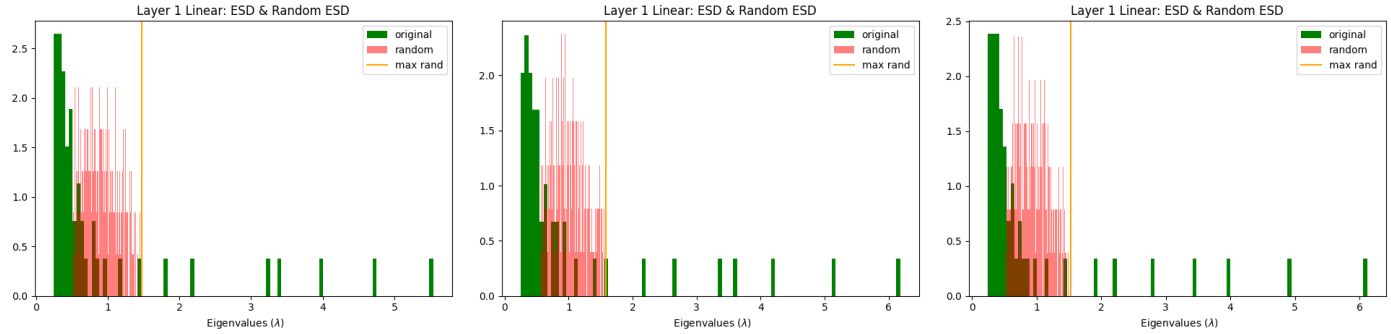

Figure 5 *WW* plots for dense and LEGCNet-FT, and LEGCNet-PT networks on MNIST data (layer 1), Plots reveal the correct training of the proposed architectures. WW plots of MNIST for random and magnitude pruning are available in Appendix

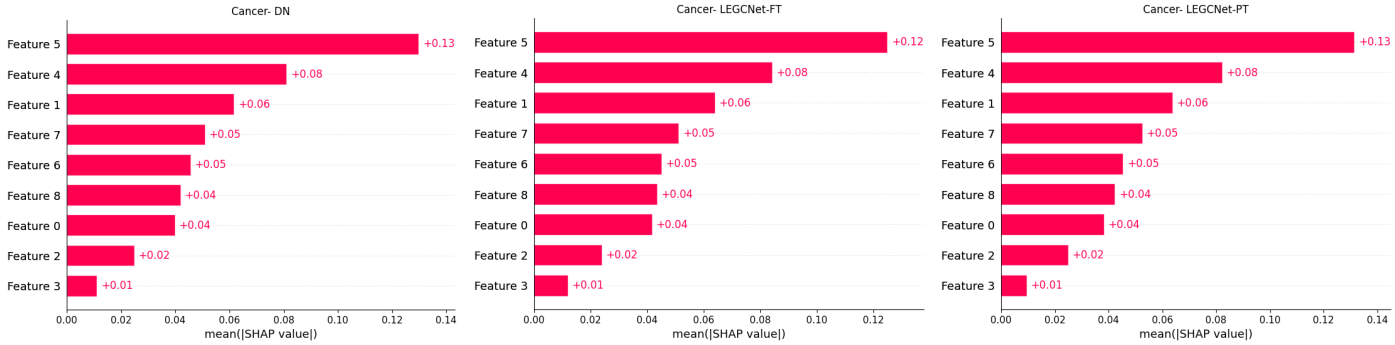

Figure 6 Shap values and feature importance computed on Cancer dataset (Banknote and Titanic plots can be seen in Appendix) for all three models - Dense Network, LEGCNet-FT and LEGCNet-PT; the feature importance for the dense network is the same as LEGCNet-FT and LEGCNet-PT

values from game theory and their related extensions. SHAP enables a comprehensive understanding of the contributions made by different features in the model's output, facilitating insightful explanations for its decision-making process. If a network is pruned according to some underlying principles, then the consistency in feature explainability is maintained before and after pruning i.e. the features that explain the outcome before pruning (fully connected, dense network) remain consistent on the pruned network.

The SHAP values computed for the proposed pruning architecture as well as for the baselines - random and magnitude - indicate that the feature importance is maintained in LEGCNet-PT and LEGCNet-FT when compared with dense (Figure 6). However, the baseline pruning methods (random, magnitude pruning) could not maintain the feature consistency as seen in the SHAP plots. Though magnitude pruning shows feature consistency for Cancer, banknote, and Titanic datasets, the random pruning could not (please refer Appendix file for these plots).

Unlike the current baselines, the percentage of pruned weights in the proposed work is significantly less. This is because only the non-causal weights are pruned, weights that play no role in impacting the loss/accuracy. LEGCNet ensures a mechanism to check which weights to prune and which ones to keep by measuring the causality between the LEs of weights and misclassification. Whether the pruned weights are large in number or small, is guided by connections that contribute to the training. Additionally, if the network is not overparameterized, the methodology righteously removes those connections that do not contribute during training. We argue that the proposed strategy is efficient and accurate, with the additional benefit of passing network fitting and explainability tests, in addition to satisfying the lottery ticket hypothesis (Tables 1 and 2 Experimental section.). One of the other salient features of LEGCNet is that, unlike other pruning methods, it does not need the dense network to be trained for the entire cycle of epochs to identify pruning candidates. Rather, such candidates are detected after a few initial epochs so that the retraining can start immediately. This is reflected in reduced FLOPs without compromising key performance indicators (Tables 1, 2). Notably, our architecture is validated for correct training via *WW* Statistics as detailed in the diagnostic section previously. We discuss more on our findings from these experiments.

How do our results align with the Lottery ticket hypothesis? Our work validates the lottery ticket hypothesis empirically, by showing the comparable (and sometimes better) performance of LEGCNet-PT and LEGCNet-FT as against dense architecture, as reflected in the Tables 1 and 2 in the Experimental section. The pruned network not only gets trained with lesser epochs than the dense, but it also uses lesser FLOPs to achieve this performance.

Is our architecture Validated for correct training via WW Statistics? As detailed in the WeightWatcher section previously, we have conducted a thorough examination of our pruning method's impact on neural network training, validating that it maintains the network's capability to encode accurate representations from the data, akin to the method without pruning. However, it is important to acknowledge that some skepticism may arise concerning the results presented in the table. Notably, we observed that the middle layer consistently exhibited signs of undertraining, irrespective of whether pruning was applied or not. This phenomenon can be at-

tributed to the architecture's design, where the first layer, due to its significantly higher number of interactions, encodes a substantial portion of the information, leaving the middle layer comparatively underutilized in terms of information encoding. Moreover, the final layer efficiently captures the essential information required for accurate classification, which has been frequently observed in deep-layer neural networks with a roughly uniform node count of around 30 nodes per layer (except the first and last layers).

It is crucial to emphasize that our primary focus in this investigation was on the effects of pruning, rather than achieving perfect classification performance. The benchmark accuracy served as an indicative measure of model performance, while the underlying motivation centered around the observation that ESD (Eigenvalue Spacing Distribution) plots do not align precisely with those of random matrices. In conclusion, while the middle layer's undertraining may raise concerns, the core contribution of this study lies in demonstrating that our pruning method effectively retains the neural network's ability to encode meaningful representations from the data, ensuring that essential information is preserved while achieving a desirable balance between accuracy and interpretability. These insights serve to shed light on the intricate interplay between pruning and neural network architectures.

How do the proposed method and architectures produce consistent feature explainability, in contrast with baseline pruning architectures? The experimental results demonstrated that the proposed pruning method exhibits notable advantages in maintaining feature importance compared to the traditional random and magnitude pruning methods. The feature consistency remained relatively stable after employing the proposed pruning technique, which was not the case for the other two methods. For random pruning, we observed that pruning 40% of the weights led to satisfactory accuracy levels for the respective datasets. Similarly, magnitude pruning, with a 25% pruning threshold based on weight magnitudes, yielded comparable benchmark accuracies. The SHAP plots provided critical insights into the behavior of feature importance during pruning. In the case of random and magnitude pruning, significant fluctuations in feature importance were evident after pruning. These fluctuations could potentially hinder the interpretability of the underlying model. However, our proposed pruning method demonstrated remarkable resilience in preserving feature importance, with minimal perturbations observed in SHAP plot patterns enabling a more interpretable and transparent pruned model. By selectively targeting weights based on a novel criterion, the proposed method ensures that crucial features are retained, thus For mission-critical tasks on edge devices such as predicting power consumption of applications (Alavani *et al.* 2023) or forecasting real-time blood glucose prediction, feature explainability on pruned networks is critical as it helps determine accurate prediction when dimensionality is a curse. *How is our method doing in comparison with baseline pruning methods?* The overall performance of our methods, when compared with the dense and baselines, shows that the Chaos-Causality framework consumes fewer epochs and fewer FLOPs (Tables 1, 2) to train the sparse network without compromising the accuracy and F1-score thus validating the Lottery ticket hypothesis. When compared with dense networks, LEGCNet-PT and LEGCNET-FT needed lesser training epochs on 6 out of 7 datasets, while the accuracy remained at par across all methods.

CONCLUSION

The main idea is to decide on a weight connection in the dense neural network, whether it should 'be pruned or not to be pruned', and this decision is governed by the presence or absence of the causal relationship between the Lyapunov exponent and the misclassification loss during the neural network training (backpropagation) for the specific connection. If the causality is established between them, it is pruned otherwise it remains unchanged during the backpropagation. Once pruned, we investigate whether the pruned network performs well or becomes worse compared with its dense counterpart. To validate this, we train the sparse network till convergence and compare its performance.

Tables 1 and 2 show that comparison. We compared the FLOPs, percentage pruned (fraction of parameters removed *100), accuracy, f1-scores, and epochs for all methods (dense, LEGCNet-FT and LEGCNet-PT). Table 1 shows the performance comparison of dense network and LEGCNet-FT. Remarkably, LEGCNet-FT achieves notable reductions in FLOPs without compromising accuracy. Furthermore, LEGCNet-FT converges significantly faster consuming a few epochs compared to the dense network. Specifically, for the Titanic, Vowel, and Cancer datasets, LEGCNet-FT achieves convergence in just half the number of epochs required by the dense network. Nonetheless, both networks achieve a similar performance level without significant differences, thus validating the lottery ticket hypothesis. Table 2 demonstrates the performance of LEGCNet-PT. It shows that LEGCNet-PT performs at par with its counterpart, in terms of convergence speed, FLOPs, accuracy, and F1 scores.

In conclusion, we emphasize that our primary focus in this investigation was on the effects of pruning, rather than achieving perfect classification performance. In the past, pruning was typically accomplished by ranking or penalizing weights based on criteria like magnitude and removing low-ranked weights before retraining the remaining ones. We formulate pruning as an optimization problem to minimize misclassifications by selecting specific weights. To pick these weights, we have introduced the concept of chaos in learning (Lyapunov Exponents) through weight updates and have used causality-based investigation to identify those causal weight connections responsible for misclassification. We proposed two architectures - Lyapunov Exponent Granger Causality driven Fully Trained Network (LEGCNet-FT) and Lyapunov Exponent Granger Causality driven Partially Trained Network (LEGCNet-PT). We compared three distinct pruning techniques: random pruning, magnitude pruning, and our pruning approach. Overall, the experimental outcomes validate the superiority of the proposed pruning method. The findings hold great promise for further advancements in network optimization and model explainability. We demonstrated that:

- By incorporating Lyapunov exponent (LE) values from weight updates and verifying the causality of LE with accuracy, one can identify certain weights in a dense network that do not significantly contribute to improving accuracy or diminishing misclassification
- Pruning these unnecessary (noncausal) weights results in a subnetwork that can achieve the same or sometimes better validation accuracy as the original dense network.
- The sub-network, once pruned, and trained from the start (offline pruning) with the same initialization as the dense network, achieves comparable, and sometimes better, training speed while reaching the same validation accuracy.
- The subnetwork, LEGCNet-PT, after pruning, does not require further tuning to match the performance of the original dense

network.

- By using LEGCNet-PT,/LEGCNet-FT to prune the weight connections, a dense over-parameterised network determines weight connections to be pruned without compromising on accuracy/F1 score or any other performance metrics.
- When compared with the classical methods like random pruning and magnitude pruning we observe that our methods perform at par and sometimes better in terms of classification accuracy on all datasets.
- The accuracy comparison of dense networks and sparse networks (using LEGCNet-PT, LEGCNet-FT) reveals that the sparse networks perform better.
- We also found via experiments that the sparse networks were trained on lesser epochs and fewer FLOPs than their dense counterpart on all datasets.
- By using SHAP, we showed that a dense network when pruned using LEGCNet-PT/LEGCNet-FT demonstrates remarkable resilience in preserving feature importance, and enables a more interpretable and transparent pruned model.

Our pruning approach is yet to be tested on baseline architectures (Resnet, Densenet), and Large Language Models and savings in carbon emission need to be computed.

Acknowledgments

Archana Mathur and Nithin Nagaraj would like to thank SERB-TARE for supporting the work under Grants TAR/2021/000206. Snehanshu Saha would like to thank the Anuradha and Prashanth Palakurthi Center for Artificial Intelligence Research (APPCAIR), SERB SURE-DST (SUR/2022/001965), ANRF-CRG (CRG/2023/003210), and the DBT-Builder project (BT/INF/22/SP42543/2021), Govt. of India for partial support.

Ethical standard

The authors have no relevant financial or non-financial interests to disclose.

Availability of data and material

Not applicable.

Conflicts of interest

The authors declare that there is no conflict of interest with respect to the publication of this article.

LITERATURE CITED

Alavani, G., J. Desai, S. Saha, and S. Sarkar, 2023 Program analysis and machine learning based approach to predict power consumption of cuda kernel. ACM Transactions on Modeling and Performance Evaluation of Computing Systems .

Balakrishnan, H. N., A. Kathpalia, S. Saha, and N. Nagaraj, 2019 Chaosnet: A chaos based artificial neural network architecture for classification. Chaos: An Interdisciplinary Journal of Nonlinear Science **29**.

Ditto, W. L. and S. Sinha, 2015 Exploiting chaos for applications. Chaos: An Interdisciplinary Journal of Nonlinear Science **25**.

Faure, P. and H. Korn, 2001 Is there chaos in the brain? i. concepts of nonlinear dynamics and methods of investigation. Comptes Rendus de l'Académie des Sciences-Series III-Sciences de la Vie **324**: 773–793.

Frankle, J. and M. Carbin, 2018 The lottery ticket hypothesis: Finding sparse, trainable neural networks. ICLR 2019 .

Granger, C. W., 1969 Investigating causal relations by econometric models and cross-spectral methods. Econometrica: journal of the Econometric Society pp. 424–438.

Hegger, R., H. Kantz, and T. Schreiber, 1998 Practical implementation of nonlinear time series methods: The tisean package. Chaos **9 2**: 413–435.

Herrmann, L. M., M. Granz, and T. Landgraf, 2022 Chaotic dynamics are intrinsic to neural network training with sgd. In *Neural Information Processing Systems*.

Kondo, M., S. Sunada, and T. Niiyama, 2021 Lyapunov exponent analysis for multilayer neural networks. Nonlinear Theory and Its Applications, IEICE .

Korn, H. and P. Faure, 2003 Is there chaos in the brain? ii. experimental evidence and related models. Comptes rendus biologies **326**: 787–840.

Kuo, D., 2005 Chaos and its computing paradigm. IEEE Potentials **24**: 13–15.

LeCun, Y., I. Kanter, and S. A. Solla, 1990 Second order properties of error surfaces: Learning time and generalization. In *NIPS 1990*.

Lee, J., S. Park, S. Mo, S. Ahn, and J. Shin, 2020 Layer-adaptive sparsity for the magnitude-based pruning. In *International Conference on Learning Representations*.

Li, G., C. Qian, C. Jiang, X. Lu, and K. Tang, 2018 Optimization based layer-wise magnitude-based pruning for dnn compression. In *International Joint Conference on Artificial Intelligence*.

Liu, S., T. Chen, X. Chen, L. Shen, D. C. Mocanu, *et al.*, 2022 The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, OpenReview.net.

Liu, Z., M. Sun, T. Zhou, G. Huang, and T. Darrell, 2018 Rethinking the value of network pruning. ArXiv **abs/1810.05270**.

Lundberg, S. M. and S.-I. Lee, 2017 A unified approach to interpreting model predictions. Red Hook, NY, USA, Curran Associates Inc.

Martin, C. H., T. Peng, and M. W. Mahoney, 2020 Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. Nature Communications **12**.

Medhat, S., H. Abdel-Galil, A. E. Aboutabl, and H. Saleh, 2023 Iterative magnitude pruning-based light-version of alexnet for skin cancer classification. Neural Computing and Applications pp. 1–16.

Mittal, D., S. Bhardwaj, M. M. Khapra, and B. Ravindran, 2018a Recovering from random pruning: On the plasticity of deep convolutional neural networks. 2018 IEEE Winter Conference on Applications of Computer Vision (WACV) pp. 848–857.

Mittal, D., S. Bhardwaj, M. M. Khapra, and B. Ravindran, 2018b Studying the plasticity in deep convolutional neural networks using random pruning. Machine Vision and Applications **30**: 203 – 216.

Mohapatra, R., S. Saha, C. A. C. Coello, A. Bhattacharya, S. S. Dhavala, *et al.*, 2022 Adaswarm: Augmenting gradient-based optimizers in deep learning with swarm intelligence. IEEE Transactions on Emerging Topics in Computational Intelligence **6**: 329–340.

Nagaraj, N. and P. G. Vaidya, 2009 Multiplexing of discrete chaotic signals in presence of noise. Chaos: An Interdisciplinary Journal of Nonlinear Science **19**.

Prandi, C., S. Mirri, S. Ferretti, and P. Salomoni, 2017 On the need of trustworthy sensing and crowdsourcing for urban accessibility in smart city. ACM Trans. Internet Technol. **18**.

Qian, X.-Y. and D. Klabjan, 2021 A probabilistic approach to neural network pruning. In *International Conference on Machine Learning*.

Rosenstein, M. T., J. J. Collins, and C. J. D. Luca, 1993 A practical method for calculating largest lyapunov exponents from small data sets. Physica D: Nonlinear Phenomena **65**: 117–134.

Saha, S., N. Nagaraj, A. Mathur, R. Yedida, and S. H. R, 2020 Evolution of novel activation functions in neural network training for astronomy data: habitability classification of exoplanets. The European Physical Journal. Special Topics **229**: 2629 – 2738.

Saleem, T. J., R. Ahuja, S. Prasad, and B. Lall, 2024 Insights into the lottery ticket hypothesis and iterative magnitude pruning. ArXiv **abs/2403.15022**.

Shen, Z., H. Yang, and S. Zhang, 2019 Nonlinear approximation via compositions. Neural networks : the official journal of the International Neural Network Society **119**: 74–84.

Sprott, J., 2013 Is chaos good for learning. Nonlinear dynamics, psychology, and life sciences **17**: 223–232.

Wolf, A., J. Swift, H. L. Swinney, and J. A. Vastano, 1985 Determining lyapunov exponents from a time series. Physica D: Nonlinear Phenomena **16**: 285–317.

Zhang, L., L. Feng, K. Chen, and C. H. Lai, 2021 Edge of chaos as a guiding principle for modern neural network training. ArXiv **abs/2107.09437**.

Zou, D., Y. Cao, D. Zhou, and Q. Gu, 2018 Stochastic gradient descent optimizes over-parameterized deep relu networks.

How to cite this article: Sahu, R., Chadha, S., Mathur, A., Nagaraj, N. and Saha, S. A Chaos-Causality Approach to Principled Pruning of Dense Neural Networks. *Chaos Theory and Applications*, 7(2), 154-165, 2025.