Towards understanding and modelling sparse training algorithms at extreme sparsity regime

Vincent van Engers
University of Twente
PO Box 217, 7500 AE Enschede
the Netherlands
v.p.g.vanengers@student.utwente.nl

ABSTRACT

Deep neural networks have proven useful in practical applications. However, many redundant connections unnecessarily inflate network size and computational complexity. Inspired by pruning in biological brains, sparse training methods such as Sparse Evolutionary Training (SET) and Accuracy based Sparse Evolutionary Training (AccSET) prove more efficient than fully connected counterparts by dynamically adding and removing connections. Contrasting to the great amount of research on deep neural networks, little research exists on the effect of varying hyperparameters and structure on the performance and behaviour of sparse neural networks. This paper investigates the influence of varying sparsity levels on the behaviour and performance of Sparse Evolutionary Training and provides new insights related to sparse training over a large sparsity horizon. This paper categorizes different levels of sparsity and defines the extreme sparse contour. It further delivers a systematic analysis of extremely sparse neural networks and a mathematical formulation of the relation between sparsity and accuracy which can estimate with great accuracy the expected accuracy of a model at a given sparsity level. The experiments in this research show that certain sparse neural networks can be trained at extreme sparsity levels. With these results, this paper contributes to the understanding of sparse training in artificial neural networks and underlines the importance of sparse neural networks to the implementation of machine learning in limited computing devices.

Keywords

Artificial Neural Network, Sparse Neural Network, Sparse Evolutionary Training, Synaptic Pruning, Accuracy based Sparse Evolutionary Training, Extreme Sparsity Regime.

1. INTRODUCTION

Deep learning has proven effective for practical applications as image recognition, natural language processing and autonomous driving [21]. Through its popularity, deep learning has become synonymous with artificial intelligence. While some results can be practically applied, the great number of connections inherent to deep neural networks necessitate great computing power and could lead to overfitting [20]. Backpropagation lies at the basis of most modern machine learning. [30] Applications of the chain rule in programming have been introduced in the 1960s and 1970s as a precursor to backpropagation [15]. Backpropagation itself has been popularized as early as 1986 [31]. Still, much machine learning literature was limited to theoretical research because of the great computational cost of neural networks and the absence of required computing capacity [34]. While computing power has greatly increased over the years, many state-of-the-art fully connected deep learning models still require great computing power with great computational cost [34]. Sparse neural networks address this large computing requirement.

Research on biological neural networks supports the concept of sparsity in artificial neural networks as it has shown that connectivity between neurons decreases with increased cortical size [17]. Similarly, research on artificial neural networks has revealed many of its connections to be redundant [11]. Subsequent techniques have been developed to increase the effectiveness and training speed of neural networks by removing redundant connections, creating sparse neural networks [26]. Initial sparse training methods start with a fully connected neural network. Subsequently, this fully connected neural network is iteratively pruned to achieve the sparse structure [14, 16, 20, 22]. Other sparse training methods initialise a sparse structure from scratch that is subsequently trained in sparse-to-sparse training[10, 12, 13, 19, 27].

Research has found that biological brains constantly add and remove connections. Contrasting, initial sparse training algorithms and many sparse-to-sparse training methods prune edges before training [26]. In 2018 Mocanu et al. [27] proposed Sparse Evolutionary Training, a sparse training model inspired by synaptic pruning in biological brains. SET constantly adds and removes edges during training. In 2020 Lapshyna [19] observed that the total number of connections in biological brains changes while the total number of edges during SET is fixed and introduced adaptive performance-based connectivity in AccSET

Different sparse training methods have been developed over the past years [5, 10, 12, 16, 22, 27]. However, little literature exists on the influence of varying hyperparameter and model characteristics on the performance of sparse neural networks [8]. Researchers have already shown great decrease in parameter count with insignificant accuracy decrease [26]. With sparsity at the core of this fairly new research field, this paper will investigate the following research question:

What is the influence of varying sparsity levels on the behaviour and performance of Sparse Evolutionary Training?

The Background section of this paper explains the basic concepts on which this paper is based. It introduces the fundamentals of the multilayer perceptron, sparse training and SET. The Related work section discusses literature most relevant to this research, discussing related papers on sparse training, Sparse Evolutionary Training, training in the extreme sparsity regime and difficulties of training sparse neural

35th Twente Student Conference on IT, July 2nd, 2021, Enschede, The Netherlands.

networks. The Method section discusses the research methods used in this paper. The Results section specifies the implementation details, presents the experiment results and provides a systematic analysis. The Conclusion section discusses the implications of the results and the relevance of this paper to research in sparse neural networks. Finally, the Future work section discusses the limitations of this paper and how the results invite further research.

Through the analysis of training Sparse Evolutionary Training at varying sparsity levels, this paper contributes:

- New insights related to the sparse training over a large sparsity horizon
- A systematic analysis of extremely sparse networks
- A mathematical formulation of the relation between sparsity and accuracy, estimating with great accuracy the expected accuracy of a model at a given sparsity level.

2. BACKGROUND

2.1 Multilayer perceptron (MLP)

Inspired by neural networks in the brain, the multilayer perceptron is an artificial neural network that can be trained to produce accurate predictions based on input data [1]. It consists of artificial neurons that are ordered in an input-, output- and one or multiple hidden layers. Neurons are connected to neurons in neighbouring layers through connections, parameters or edges. The total number of edges is also referred to as parameter count. Every edge is associated with a weight value that modifies values as they propagate from one neuron to the next. Weights are initially set randomly and are subsequently trained to collectively make accurate predictions on given data. Every artificial neuron has an activation function that transforms its output number to a normalized value [5]. Figure 1 shows the structure of artificial neurons and multilayer perceptrons.

Unless weights are set manually through a priori knowledge, MLPs are trained through supervised-, unsupervised-, or reinforcement learning. Supervised learning involves the adjustment of weights based on the prediction errors of the MLP. The adjustment of the weights is carried out by backpropagation [30]. In unsupervised learning, there is no a priori set of classifications. The neural network is supposed to discover statistically significant features based on the input data. In reinforcement learning, the system learns through trial and error how situations should be mapped to actions. A reward function provides feedback on the appropriateness of an action in each situation. See Artificial Neural Networks by Abraham [1] for further explanation of MLPs.

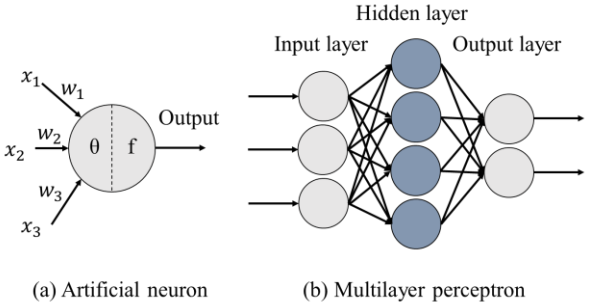


Figure 1: *Artificial neuron (a) and multilayer perceptron (b)*

2.2 Sparse neural networks

In conventional MLPs, a single neuron is connected to all neurons in neighbouring layers. However, many edges in these fully connected MLPs are associated with near-zero weights and can be removed to greatly decrease hardware impact while limiting accuracy decrease [10, 12, 16, 20]. MLPs with at least one edge missing between layers are called sparse neural networks. Figure 2 shows the difference between fully connected- and sparsely connected layers.

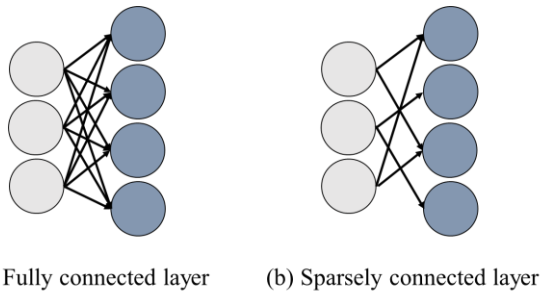


Figure 2: *Fully connected layers (a) vs sparsely connected layers (b)*

2.3 Sparse Evolutionary Training (SET)

As will be discussed in Section 3.1 and 3.2.1, many sparse training methods initially reduce the number of connections once and train afterwards [9, 14, 16, 20, 22, 25]. Contrasting, biological brains constantly remove and add connections [5]. SET [24, 27] is a sparse training method inspired by this behaviour in biological brains. SET constantly adds and removes edges during training. The method works as follows:

1. Initialise a random sparse neural network with sparsity level ε.
2. Update weights.
3. Remove a fixed fraction ζ of edges with the smallest positive and the largest negative weights.
4. Add the same number of edges ζ and randomize their weights.
5. Repeat steps 2 – 4 until the desired accuracy is achieved. Omit step 4 on the last training epoch.

3. RELATED WORK

Sparse neural networks have proven effective in greatly decreasing parameter count while maintaining high accuracy levels [10, 12, 16, 20]. Consequently, the subject has seen a great increase in published literature over the past years [33]. Various methods have been proposed to optimize pruning techniques and model performance. This section will discuss the most relevant recent developments in sparse neural training.

3.1 Dense-to-sparse training

In a fully connected multilayer perceptron, every artificial neuron is connected to all neurons in the neighbouring layers. For every layer L_n except the last layer, Ln x Ln+1 edges are connecting it to the next layer and need to be updated in every epoch. Consequently, there has been much incentive to reduce the number of edges that need to be updated and much research has been focused on pruning.

The term pruning has been introduced in 1989 by Mozer and Smolensky [28] and in 1991 by LeCun et al. [20] in Optimal Brain Damage (OBD). OBD was the first sparse training method using dense-to-sparse training. It consists of an initial

dense training phase where a regular fully connected neural network is trained and where it becomes evident which connections are important to performance. Afterwards, the sparse training phase is used to *prune* the connections that are unessential towards network performance. These training- and pruning phases are repeated for several steps p until an optimum between parameter count and performance is found. In 2015 Han et al. [16] proposed a three-step pruning method that refined the former method for deep neural networks. On the ImageNet dataset, compared to denser connected counterparts AlexNet and VGG-16, the training method reduces the parameter count by 9x and 13x respectively and greatly reduces the percentage of near-zero weights.

While these initial pruning methods have been able to drastically decrease parameter count, the smaller architectures resulting from pruning are more difficult to train from the start and reach lower accuracy than the original networks. Answering this problem, in 2019, Frankle and Carbin [14] introduced the Lottery Ticket Hypothesis. It states that if the remaining weights after pruning are set to their original initialisation value, this subnetwork can be trained to match the test accuracy of the original network with at most the same number of iterations. However, if the weights of the same subnetwork are set to random values, the performance achieved is significantly lower. The hypothesis has shown that if the remaining weights are set to their initial value, the number of successive training steps can be reduced to p=2.

Alternatively, in 2019 Lee et al. [22] developed Single-Shot-Network Pruning (SNIP), a sparse training method that prunes a network once before training. Connections are pruned based on their influence on the loss function. The resulting sparse network is subsequently trained without additional pruning. On the CIFAR-10 dataset, compared to AlexNet, SNIP reduces the parameter count by 10x with less than one percent increase in error.

3.2 Sparse-to-sparse training

Dense-to-sparse training has proven effective in reducing parameter count while achieving high performance. In addition to pruning fully connected layers in dense-to-sparse training, much literature has started focusing on sparse-to-sparse training. These are neural networks that start with a sparse architecture and are subsequently further pruned and trained. Sparse-to-sparse training is often referred to as *sparse training.*

3.2.1 Static sparse connectivity

Various methods exist to initialize the fixed sparsity structure before training. In 2016 Mocanu et al. introduced the first model training sparse neural networks from scratch instead of pruning a fully connected and pre-trained neural network [25]. The proposed Complex Boltzmann Machines (XBM) which nodes are not fully connected to all neighbouring nodes are an adaptation of Restricted Boltzmann Machines (RBM) [9]. The initial sparse topological structure before training is achieved by using varying concepts from graph theory, network science and data statistics. The XBM outperforms many previous sparse training methods and achieves similar performance as RBM with 40 times fewer weights.

In 2017, Bourely et al. [4] showed that random initialization of a fixed sparse structure can achieve higher accuracy than fully connected neural networks. Another approach is the initialization of a fixed sparse structure through pre-training. Inspired by the previously mentioned Lottery Ticket Hypothesis, in 2020, You et al. [36] suggested that "winning tickets" could be drawn very early in training instead of pruning a fully trained artificial neural network. Compared to the original Lottery Ticket Hypothesis, these Early Bird tickets allow for training with very low training cost.

These training methods have shown effective in decreasing parameter count while maintaining high accuracy. However, the main disadvantage of sparse neural networks with static sparse connectivity is that the optimal sparse topology has to be discovered and designed manually.

3.2.2 Dynamic sparse connectivity

Addressing this problem, in 2018 Dai et al. [10] proposed NeST, a sparse training method that initializes a random sparse neural network. It subsequently alternates between adding connections that can quickly reduce the value of the loss function and removing connections whose weight is below a pre-defined threshold. On the ImageNet dataset, compared to AlexNet, NeST reduces the parameter count by 15.7x with an insignificant change in error rate.

Another method, Sparse Momentum, published in 2019 by Dettmers [12] prunes weights by magnitude, redistributes weights across layers dependent on the mean momentum[1] magnitude of already existing weights and grows new weights according to momentum magnitude of zero-valued weights. On the CIFAR-10 dataset, compared to AlexNet-s and Alexnet-b, Sparse Momentum reduces the parameter count by 10x with a 1,7% to 2% increase in error.

Similar to pruning, dynamic sparse connectivity has been inspired by the biological brain. Unlike static sparse training methods, it has been shown that the biological constantly prunes redundant- and grows new connections [5].

In 2017 and 2018, Mocanu et al. [24, 27] proposed SET that adds and removes a static percentage of edges during training and keeps the total number of edges fixed. SET reduces the number of edges required quadratically in bipartite layers and achieves greater accuracy than fully connected MLPs. Compared to Sparse Momentum [12], SET performs with slightly lower accuracy on the CIFAR10 and MNIST database with a WRN-28-2 and LeNet 300-100 model respectively.

Inspired by biological neural networks and SET, in 2020 Lapshyna [19] proposed AccSET that dynamically adds edges during training based on accuracy. The total number of edges is dynamic. Compared to SET, AccSET performs in experiments on the Fashion-MNIST, CIFAR10 and HIGSS dataset with slightly worse accuracy and 65.3%, 51.6% and 58.6% decrease in the number of connections [19].

Continuing on the Lottery Ticket Hypothesis, in 2020 Evci et al. [13] proposed Rigging the Lottery: Making All Tickets Winners. Training sparse networks with effective initial values is a quick process. However, the original research in which the Lottery Ticket Hypothesis was proposed, requires to first train a fully connected neural network to convergence to discover the sparse topology. Evci et al. propose Rigging the Lottery (RigL), where a randomly initialized neural network is used to achieve similar performance without needing to discover an effective topology (the "winning ticket").

3.3 Extreme sparsity in training

With the success of sparse training in the aforementioned literature in developing highly performant sparse neural

[1] See [29] for definitions.

networks, researchers have started investigating how to maximize the sparsity in sparse training.

Cho et al. [6] observed a general trade-off in the literature between computation, sensitivity to hyperparameters and test performance. In 2020 Cho et al. [6] suggested Extremely Sparse Pruned Networks (ESPN), a sparse training method inspired by SNIP. While SNIP uses a single shot estimator, ESPN uses standard iterative gradient updating to learn a sparse mask. ESPN achieves extremely high sparsity levels (> 99%) on various datasets with higher test accuracy compared to other sparse training methods such as SNIP.

Alternatively, Yu et al. [35] propose a joint pruning and quantization method to achieve extremely high sparsity (~99%). The quantization phase uses the sparse model from the pruning phase and has a neglectable impact on computational performance. The quantization phase transforms weights into powers of two. It allows the model to use shifters instead of multipliers and further reduces hardware impact. The model achieves almost 99% memory reduction on stereo depth estimation and a 99.7% hardware cost reduction.

3.4 Truly sparse neural networks

The decrease in connections in sparse neural networks reduces required computing capacity and increases efficiency in training [10, 12, 16, 20]. However, at the time of writing this paper, for most implementations, these improvements are theoretical. As Curci et al. [8] state, most modern deep neural networks frameworks are optimized for dense matrix multiplications on graphics processing units (GPUs). The sparsity is only simulated by training a binary weight mask. Ultimately these sparse neural networks are trained as dense neural networks and merely provide insight into the capacity of true sparse neural networks. Currently, the only hardware exception is the 2020 NVIDIA A100 GPU [18], supporting a sparsity level of 50%.

Curci et al. [8] further mention that next to the supporting hardware architecture, most literature on various aspects of neural networks such as optimizers and activation functions are focussed on deep neural networks and subsequently argue that these subjects should be revisited for sparse neural networks.

Addressing the problem of spare matrix multiplications, Liu et al. [23] proposed in 2021 a pure Python implementation that allows SET to be trained with 1.000.000 neurons on a standard laptop without a GPU. This is two orders of magnitude larger than any MLP trained on commodity hardware. The implementation uses various sparse data structure from the Python library SciPy [32]: compressed sparse rows [2], linked lists, coordinate list and dictionaries of keys.

4. METHOD

In this section, the method and devised strategy for the execution of the research are discussed. Three steps have been performed in the execution of experiments for this research. In the first step, SET has been trained with various levels of sparsity. In the second step, the results of the second step have been used to analyse the influence of sparsity on the performance of SET and have been used to categorized different sparsity levels. In the third step, it has been determined what mathematical function best describes the relation between sparsity and accuracy.

4.1 Step 1: Mapping sparsity, accuracy and structure

As indicated in Section 2.3, the sparsity in SET is determined by two hyperparameters: ε and ζ. ε determines the initial sparsity of the neural network and ζ determines the sparsity during training. Values for ε and ζ lie in the interval [0, 1] and represent the percentage of sparsity. Regions within the initial interval that appear to require more detailed investigation have been trained with successfully smaller step sizes until the accuracy of sufficient precise sparsity levels has been achieved. The accuracy for each sparsity level ζ is plotted with accuracy along the vertical axis and sparsity along the horizontal axis.

Further, to better understand the influence of different sparsity levels on the neural network itself, these same intervals have been used to visualize how different sparsity levels influence the structure of the sparse neural network. Specifically:

- The number of connections from each neuron in the first layer to all the neurons in the first hidden layer.
- The sum of weights associated with the connections from each neuron in the first layer to all the neurons in the first hidden layer.

4.2 Step 2: Classifying sparsity

The results of Step 1 have been used to investigate the influence of sparsity on the structure and performance of Sparse Evolutionary Training. Having mapped the sparsity to the accuracy and the structure of the network, different sparsity intervals have been classified depending on their effect on accuracy and structure.

4.3 Step 3: Mathematical modelling

The form of the plotted graph as described in Step 1 has been used to estimate the type of function that most accurately models the relation between sparsity and accuracy. Parameters have been used to shift and scale the function to the most appropriate proportions and allow for future modification that might be required with different hyperparameters or datasets. Parameters whose function can be directly interpreted in terms of their relation to characteristics of the dataset, model or results have been defined as such and set to the appropriate value. Other parameters have been given placeholder names and their values have been determined by multivariate gradient descent.

5. RESULTS

The following section discusses the conducted experiments as described in the previous section and their subsequent results. First, the implementation details of the SET algorithm are stated and the figures and tables used to display the results are explained. Second, the training results are analysed and used to categorize different sparsity levels. Third, a mathematical

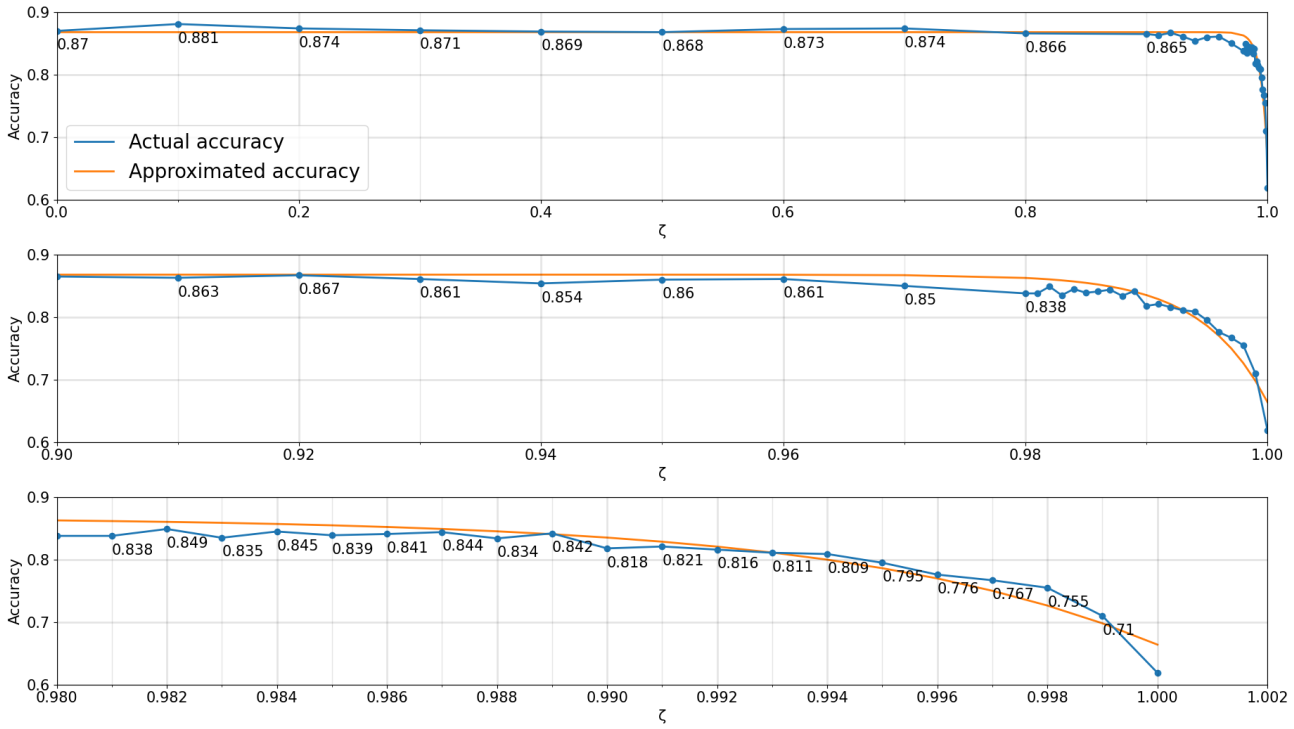
Algorithm 1: Weight pruning-regrowing cycle - Implementation II

Input : Sparse Weight Matrix (W)
Output: Sparse Weight Matrix with random weights added

%Removal of small weights
Extract values (V), row (R) and column indices (C) of the non-zeros from W
Find the maximum negative value (V_{neg}) and the minimum positive value (V_{pos})
Find the index i_{zero} of the values (V) which are bigger than V_{neg} and smaller than V_{pos}
Delete the indices of V, R and C corresponding to the index i_{zero}
N = length(i_{zero})
%Addition of random weights
Create a list of arrays (L_{old}) with the remaining elements after removing: $L_{old} = [R, C]$
while $N > 0$ **do**
    I = array of N randomly chosen from 1 to rows of W
    J = array of N randomly chosen from 1 to columns of W
    Create list (L_{new}) of arrays with k elements: $L_{new} = [I, J]$
    Remove duplicate elements from $Lnew$
    Remove elements from L_{new} in common with L_{old}
    N=N-length(L_{new})
    L_{old} = append (L_{old}, L_{new})
    Clear L_{new};
end
Append N random values to V: $V_{new} = append(V, rand(size = (1, N)))$
Unzip 1st and 2nd elements of L_{old}: $R_{new} = L_{new}(:, 1), C_{new} = L_{new}(:, 2)$
Use COO format to update the W: $W = COO(V_{new}, (R_{new}, C_{new}))$

Algorithm 1: *Weight pruning-regrowing cycle – Implementation II from Liu et al. [23]*

Figure 3: *Accuracy of SET trained on Fashion-MNIST for different ζ after 1000 epochs and approximated accuracy with intervals [0, 9] with stepsize 0.1, [0.91, 0.98] with stepsize 0.01 and [0.98, 1] with stepsize 0.001.*



ζ	0.000	0.100	0.200	0.300	0.400	0.500	0.600	0.700	0.800	0.900	0.910	0.920	0.930
Actual accuracy	0.870	0.881	0.874	0.871	0.869	0.868	0.873	0.874	0.866	0.865	0.863	0.867	0.861
Approximated accuracy	0.868	0.868	0.868	0.868	0.868	0.868	0.868	0.868	0.868	0.868	0.868	0.868	0.868
ζ	0.940	0.950	0.960	0.970	0.980	0.981	0.982	0.983	0.984	0.985	0.986	0.987	0.988
Actual accuracy	0.854	0.860	0.861	0.850	0.838	0.838	0.849	0.835	0.845	0.839	0.841	0.844	0.834
Approximated accuracy	0.868	0.868	0.868	0.867	0.862	0.861	0.860	0.859	0.857	0.855	0.852	0.849	0.845
ζ	0.989	0.990	0.991	0.992	0.993	0.994	0.995	0.996	0.997	0.998	0.999	1.000	
Actual accuracy	0.842	0.818	0.821	0.816	0.811	0.809	0.795	0.776	0.767	0.755	0.710	0.619	
Approximated accuracy	0.840	0.835	0.828	0.821	0.811	0.800	0.786	0.770	0.750	0.726	0.698	0.664	

Table 1: *Accuracy of SET trained on Fashion-MNIST for different ζ after 1000 epochs and approximated accuracy with intervals [0, 9] with stepsize 0.1, [0.91, 0.98] with stepsize 0.01 and [0.98, 1] with stepsize 0.001.*

function is devised that models the relation between sparsity and accuracy.

5.1 Implementation details

The SET algorithm was trained using the implementation from Selima Curci [7] in Python and SciPy as proposed by Liu et al. [23] mentioned in Section 3.4. The model initializes an Erdős–Rényi sparse weight masks with a uniform distribution. The sparse weight matrices are transferred to three vectors representing a compressed sparse row using coordinate lists. During the feed-forward- and backpropagation step, the weights are stored in the compressed sparse row format. The implementation of the pruning and the regrowing cycle can be seen in Algorithm 1.

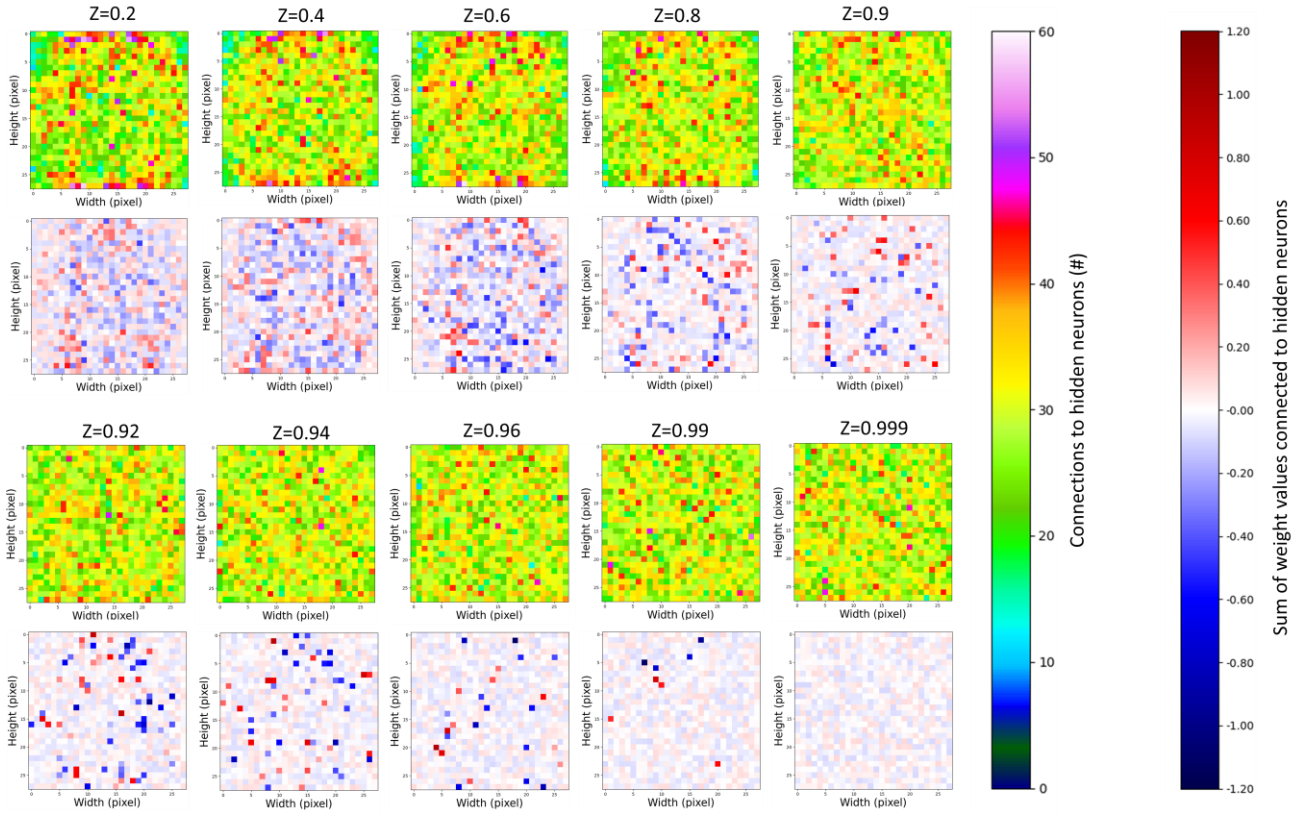
Hyperparameter	Value	Hyperparameter	Value
Learning rate	0.05	Batch size	40
Optimiser	SGD	ε	13
Momentum	0.9	Loss	Categorical
Activation Function	ReLU	Dropout rate	0.2

Table 2: *Hyperparameters used in training SET on Fashion-MNIST*

The neural network contains an input-, three hidden and one output layer. The input layer has 784 nodes, corresponding to pixels in the images of the dataset; the hidden layers contain 1000 nodes each; the output layer has 10 nodes, corresponding to the 10 different classes in the dataset. The hyperparameters can be found in Table 2. For each sparsity level ζ, the SET algorithm has been trained on Fashion-MNIST for 1000 epochs.

The experiment has been conducted as described in Section 4.1:Step 1: The algorithm has been trained three times with successively decreasing intervals and stepsizes. The SET algorithm was initially trained on ζ values in the (0,1) interval with stepsize 0.1. The accuracy remained practically unaffected for values in the (0,0.9) interval. The stepsize of 0.1 was too large to clearly show the dropoff point. Subsequent training with values in the (0,1) interval with stepsize 0.01 showed that a significant decrease started around ζ=0.98. Even further subsequent training with values in the (0.98,1) interval with stepsize 0.001 showed that accuracy starts decreasing rapidly after ζ=0.987. These results will be interpreted in Section 5.2 and Section 5.3.

Figure 4: *Heatmaps after SET trained on Fashion-MNIST for 1000 epochs with ζ={0.2, 0.4, 0.6, 0.9, 0.95, 0.96, 0.995, 1}*



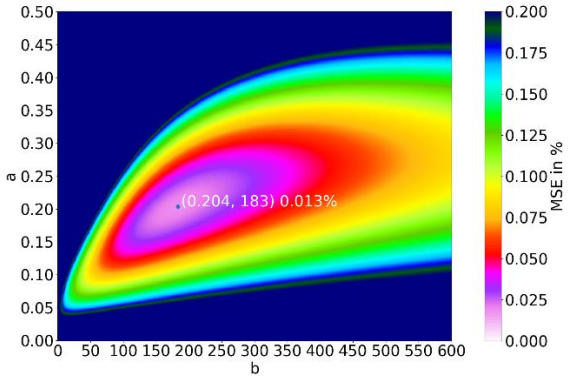


Figure 5: *Heatmap showing the MSE of Formula 2 with different values for a and b.*

In all tables and figures, accuracy is defined as the highest level of accuracy that has been achieved in any of the 1000 epochs of training on the test set.

Table 1 shows the results for different ζ values. Figure 3 graphs these results onto three plots with various zoom levels corresponding to the different intervals and stepsizes. The approximated accuracy is plotted by a mathematical function that will be elaborated in section 5.2.

Figure 4 shows heatmaps for different ζ values. The 784 pixels in the heatmap correspond to those in the dataset images and the 784 neurons in the input layer. The heatmaps in the first and third row show how many neurons are connected from that neuron in the input layer to the neurons in the first hidden layer. The heatmaps in second and fourth row show the sum of the weights of the connections from that neuron in the input layer to the neurons in the first hidden layer. In short, Figure 4 summarizes the number of neuron connections (first and third row) and the summed weight of the neuron connections (second and fourth row).

5.2 Extreme sparsity

The images in the first and third row in Figure 4 show that at lower sparsity (ζ={0.2, 0.4, 0.6}), the number of connections to the input layer are lined up to an abstract form of all images in the dataset. Because the images in the dataset do not have a background, a clear distinction is visible in the heatmaps between fore- and background. As ζ increases, the correspondence between the image and the number of connections per pixel disappears. At ζ=0.96, all visual correspondence has ceased. It appears that with such high levels of sparsity, the neural network cannot develop into a stable structure. As the number of connections cannot change in SET, the distribution of connections appears to become random.

The images in the second and fourth row show that for ζ={0.2, 0.4, 0.6} a similar structure appears to those in the images in the first and third row. Again, at higher sparsity levels, the structure starts to fade. Unlike the number of connections, the total weight of all connections can change. The heatmaps show that as the sparsity increases, most neurons are attributed with small weights and remaining neurons are associated with larger weight values. Again, the sparsity at which the correspondence to the images ceases lies at around ζ=0.96. In other research, SET and it's comparable variant AccSET use ζ=0.3 [19][27]. Table 2 and Figure 3 show that sparsity can be increased to

ζ=0.96 without any decrease in accuracy. The decrease of accuracy at ζ=0.96 corresponds to the cessation of a stable structure in the neural network at also ζ=0.96.

SET shows to be effective at high levels of sparsity. However, there appears to be a transition from normal sparsity, where the neural network operates effectively, to what can be called 'extreme sparsity', where the neural network is unable to develop a stable structure and starts losing accuracy. For this dataset and configuration of SET, the transition point or extreme sparsity contour (ESC) appears to lie at ζ=0.96.

5.3 Sparsity dependent accuracy modelling

The first plot in Figure 3 displays how accuracy changes with varying ζ in the (0, 1) interval. The long plateau at the interval (0, 0.96) and stark decrease at the interval (0.96, 1) hint at an exponential relationship. It appears that the standard exponential form e^x can be modified to conform to the first plot in Figure 3. The relation between accuracy and ζ can thus be modelled as follows:

$$acc(\zeta) = -0.204e^{183(\zeta-1)} + 0.868 \quad (1)$$

With a Mean Square Error (MSE) of 0.013%, this equation models the relationship between sparsity and accuracy of the sparse neural network with very high accuracy. Table 2 and Figure 3 shows how the approximated values of this formula plot against the actual accuracy achieved by the sparse neural network. Modelling the effect of ζ on the accuracy, this function can aid in the hyperparameter selection process.

The formula has been optimized for training SET on Fashion-MNIST with the hyperparameters from Table 1. For other datasets and hyperparameter configurations, the formula might not be suitable and needs to be adjusted. As described in Section 4.3, the function can be parameterized such that it is adaptable to any sparse neural network with a similar exponential relationship between ζ and accuracy. The general function is as follows:

$$acc(\zeta)_{a,b,acc_p} = -ae^{b(\zeta-1)} + acc_p \quad (2)$$

Here, acc_p is the mean accuracy level of the plateau of the exponential function before the extreme sparsity line:

$$acc_p = \frac{\sum_{\zeta=0}^{ESC} acc_r(\zeta)}{N_\zeta} \quad (3)$$

Here, $Acc_r(\zeta)$ is defined as mapping the accuracy after training the sparse neural network on a dataset to ζ. Again ESC is the extreme sparsity contour. N_ζ is defined as the total number of zeta values that are being summed.

In Formula (2), a and b can be altered to adjust the angle of the function and move the curve to the most appropriate point. Optimal values for a and b can be found by using multivariate gradient descent. Figure 5 shows how different values for a and b influence the MSE of the function.

6. CONCLUSION

Inspired by SET and AccSET, this paper researches how training Sparse Evolutionary Training at extreme sparsity regime influences testing accuracy and answers the following research question:

What is the influence of varying sparsity levels on the behaviour and performance of Sparse Evolutionary Training?

In answering this question, this paper delivers

- New insights related to the sparse training over a large sparsity horizon
- A systematic analysis of extremely sparse networks
- A mathematical formulation of the relation between sparsity and accuracy, estimating with very good accuracy the expected accuracy of a model at a given sparsity level.

In this research, it has been shown that SET can be trained on Fashion-MNIST with training sparsity levels as high as 96% without losing accuracy. The accuracy plateaus before this point and roughly exponentially declines afterwards. The plateau and decline in accuracy correspond to the existence and cessation of correspondence between the number and the summed weights of the connections between input and the first hidden layer and the abstraction of the images in the dataset. This transition point between normal- and extreme sparsity is defined as the 'extreme sparsity contour'. Furthermore, an exponential function has been developed that models the influence of training sparsity on the testing accuracy with an MSE of 0.013%. The parameters can be adjusted to conform to varying hyperparameters and neural network structures. This function can aid those working on sparse neural networks in determining what sparsity level might be appropriate for a given experiment or implementation.

Sparse Neural Networks have proven effective in greatly reducing parameter count while maintaining high accuracy. With its systematic analysis of sparsity in Sparse Evolutionary Training, this paper contributes to the understanding of sparse neural networks. It underlines the effectiveness of pruning in decreasing parameter count and therewith the importance of sparse neural networks towards the implementation of machine learning in limited computing devices.

7. FUTURE WORK

It has been shown that SET can be trained up to extremely high sparsity levels without a decrease in accuracy. The relation between sparsity and accuracy has shown to be exponential and can be estimated with very high accuracy by a mathematical model. Similar to these results, research by Cho et al. [6] shows a similar exponential relationship between sparsity and accuracy for ESPN, SNIP and other sparse training methods. For ESPN the ESC appears to also lie around ζ=0.96. Further research might validate the broader relevance of ESC.

The SET algorithm has been trained with hyperparameters as shown in Table 1 on the Fashion-MNIST dataset. To confirm the generalizability of the results in this paper, further research should be conducted with varying hyperparameters, datasets and sparse training methods. Primary considerations for such research are the confirmation of an exponential relation between sparsity and accuracy, and the confirmation that the accuracy starts decreasing concurrently to the cessation of a correspondence between the number and the summed weights of the connections between input and the first hidden layer. If these primary concerns are confirmed, research might focus on secondary concerns. Currently, the parameters of $acc(\zeta)$ can only be found after initial training. In further research, a model might be developed describing the relation between these parameters and the characteristics of datasets and training methods. It will allow researchers and general users to use the function to determine appropriate sparsity levels for their training.

8. ACKNOWLEDGEMENT

My sincere gratitude goes to my supervisor Elena Mocanu. Her valuable feedback and great enthusiasm during the entire process have been of invaluable support in the development of this research.

9. REFERENCES

[1] Abraham, A. 2005. Artificial Neural Networks. In *Handbook of measuring system design*.

[2] Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C. and Van der Vorst, H. 1994. In *Templates for the solution of linear systems: building blocks for iterative methods*. Society for Industrial and Applied Mathematics.

[3] Bishop, C.M. 1995. Neural networks for pattern recognition. Oxford university press.

[4] Bourely, A., Boueri, J.P. and Choromonski, K. 2017. Sparse neural networks topologies. *arXiv preprint arXiv:1706.05683*.

[5] Chechik, G., Meilijson, I. and Ruppin, E. 1998. Synaptic pruning in development: a computational account. In *Neural computation*, *10*(7), pp.1759-1777.

[6] Cho, M., Joshi, A. and Hegde, C. 2020. ESPN: Extremely Sparse Pruned Networks. *arXiv preprint arXiv:2006.15741*.

[7] Curci, S. 2020. *SelimaC/Tutorial-SCADS-Summer-School-2020-Scalable-Deep-Learning*. [online] GitHub. Available at: <https://github.com/SelimaC/Tutorial-SCADS-Summer-School-2020-Scalable-Deep-Learning/blob/master/set_mlp.py> [Accessed 26 June 2021].

[8] Curci, S. Mocanu, D.C. and Pechenizkiyi, M., 2021. Truly Sparse Neural Networks at Scale. arXiv preprint arXiv:2102.01732.

[9] Fischer, A. and Igel, C. 2012, September. An introduction to restricted Boltzmann machines. In *Iberoamerican congress on pattern recognition* (pp. 14-36). Springer, Berlin, Heidelberg

[10] Dai, X., Yin, H. and Jha, N.K. 2019. NeST: A neural network synthesis tool based on a grow-and-prune paradigm. In *IEEE Transactions on Computers*, *68*(10), pp.1487-1497.

[11] Denil, M. Shakibi, B., Dinh, L., Ranzato, M.A. and de Freitas, N., 2013, December. Predicting parameters in deep learning. In *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2* (pp. 2148-2156).

[12] Dettmers, T. and Zettlemoyer, L. 2019. Sparse networks from scratch: Faster training without losing performance. In *International Conference on Learning Representations.* 2020.

[13] Evci, U., Gale, T., Menick, J. Castro, P.S. and Elsen, E., 2020, November. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning* (pp. 2943-2952). PMLR.

[14] Frankle, J. and Carbin, M. 2018, September. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations.* 2018.

[15] Goodfellow, I., Bengio, Y. and Courville, A. 2016. 6.5 back-propagation and other differentiation algorithms. In *Deep Learning, pp.200-220.*

[16] Han, S., Pool, J., Tran, J. and Dally, W.J. 2015. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1* (pp. 1135–1143).

[17] Herculano-Houzel, S., Mota, B., Wong, P. and Kaas, J.H. 2010. Connectivity-driven white matter scaling and folding in primate cerebral cortex. In *Proceedings of the National Academy of Sciences*, *107*(44), pp.19008-19013.

[18] Jeff Pool. Accelerating sparsity in the Nvidia ampere architecture 2020. URL https://developer.download.nvidia.com/video/gputechconf/gtc/2020/presentations/s22085-accelerating-sparsity-in-the-nvidia-amperearchitecture%E2%80%8B.pdf.

[19] Lapshyna, V. 2020. Sparse artificial neural networks: Adaptive performance-based connectivity inspired by human-brain processes. *Bachelor's thesis, University of Twente.*

[20] LeCun, Y. Denker, J.S. and Solla, S.A., 1990. *Optimal brain damage*. In Advances in neural information processing systems (pp. 598-605).

[21] LeCun, Y. Bengio, Y. and Hinton, G., 2015. Deep learning. In *Nature*, *521*(7553), pp.436-444.

[22] Lee, N. Ajanthan, T. and Torr, P.H., 2019, May. SNIP: Single-shot network pruning based on connection sensitivity. *In International Conference on Learning Representations.* 2019.

[23] Liu, S., Mocanu, D.C., Matavalam, A.R.R., Pei, Y. and Pechenizkiy, M. 2021. Sparse evolutionary deep learning with over one million artificial neurons on commodity hardware. *Neural Computing and Applications*, *33*(7), pp.2589-2604.

[24] Mocanu, D.C. 2017. Network computations in artificial intelligence. *Ph.D. thesis, Technische Universiteit Eindhoven.*

[25] Mocanu, D.C., Mocanu, E., Nguyen, P.H., Gibescu, M. and Liotta, A. 2016. A topological insight into Restricted Boltzmann Machines. In *Machine Learning*, *104*(2), pp.243-270.

[26] Monacu, D.C., Monacu, E., Pinto, T., Curci, S., Nguyen, P., Gibescu, M., Ernst, D. and Vale, Z. 2021. Sparse Training Theory for Scalable and Efficient Agents. In *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems-Blue Sky Ideas Track.*

[27] Mocanu, D.C., Mocanu, E., Stone, P., Nguyen, P.H., Gibescu, M. and Liotta, A. 2018. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. In *Nature communications*, *9*(1), pp.1-12.

[28] Mozer, M.C. and Smolensky, P. 1989. Using relevance to reduce network size automatically. *Connection Science*, *1*(1), pp.3-16.

[29] Ruder, S. 2016. An overview of gradient descent optimization algorithms. . In *International Conference on Learning Representations.* 2019.

[30] Rumelhart, D.E., Durbin, R., Golden, R. and Chauvin, Y. 1995. Backpropagation: The basic theory. In *Backpropagation: Theory, architectures and applications*, pp.1-34.

[31] Rumelhart, D.E., Hinton, G.E. and Williams, R.J. 1986. Learning representations by back-propagating errors. In N*ature*, *323*(6088), pp.533-536.

[32] Scipy.org. 2021. *SciPy.org — SciPy.org*. [online] Available at: <https://www.scipy.org/> [Accessed 26 June 2021].

[33] Souza, L. 2021. *The Case For Sparsity in Neural Networks, Part 2: Dynamic Sparsity*. [online] Numenta. Available at: <https://numenta.com/blog/2020/10/30/case-for-sparsity-in-neural-networks-part-2-dynamic-sparsity> [Accessed 25 June 2021].

[34] Thompson, N.C., Greenewald, K., Lee, K. and Manso, G.F. 2020. The computational limits of deep learning. In *International Conference on Learning Representations.* 2017.

[35] Yu, P.H., Wu, S.S., Klopp, J.P., Chen, L.G. and Chien, S.Y. 2020. Joint Pruning & Quantization for Extremely Sparse Neural Networks. In *International Conference on Learning Representations.* 2020.

[36] You, H., Li, C., Xu, P., Fu, Y., Wang, Y., Chen, X., Baraniuk, R.G., Wang, Z. and Lin, Y. 2019, September. Drawing Early-Bird Tickets: Toward More Efficient Training of Deep Networks. In *International Conference on Learning Representations*.