Robustness of sparse MLPs for supervised feature selection

Neil Kichler
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
n.j.kichler@student.utwente.nl

ABSTRACT

Deep Neural Networks have seen great success yet require increasingly higher dimensional data to be applied successfully. To reduce the ever-increasing computational, energy and memory requirements, the concept of sparsity has emerged as a leading approach. Sparse-to-sparse training methods allow training and inference on more resource-limited devices. It has been hinted in previous work (SET [39] and RigL [13]) that such methods could be applied to feature selection since they may implicitly encode the input neuron strengths during training. However, a proper investigation of this potential idea has not taken place in the domain of supervised feature selection. This paper develops a method for supervised feature selection using Sparse Evolutionary Training applied to Multi-Layer Perceptrons (SET-MLP). The focus is on investigating the robustness of this feature selection mechanism to changes in the topology of SET-MLPs. We develop and perform an experimentally driven analysis on some prominent datasets to evaluate the generalizability, initialization dependence and similarity of the underlying networks of the feature selection process. We find for the selected datasets that SET-MLP produces similar feature selections for different underlying network topologies and can recover from bad initialization. This work provides a basis for understanding whether supervised feature selection using sparse training methods are robust to topological changes. The problem addressed can have further implications in understanding sparse training, given that it visualizes some aspects of the random exploratory nature of these methods. Furthermore, it discusses the potential viability of sparse-to-sparse training methods for supervised feature selection.

Keywords

Robustness, Supervised Feature Selection, Sparse Neural Networks, Sparse Topology, Network Topology Distance

1. INTRODUCTION

Over the past decade, Deep Neural Networks (DNN) have become the de-facto standard for image classification, machine translation, natural language processing, and many more domains [30]. They tackle increasingly difficult problems by increasing the size of the models and their parameters [2, 47, 24]. This improved state-of-the-art performance, however, comes at a cost. Most modern DNNs use many dense layers but reach fundamental computational constraints and large memory footprints, often requiring many TPU/GPUs for training and inference [44, 28]. Besides, many deep learning models are prone to overparameterization [10, 11] and memorization [50, 25] in the search for generalization. Overparameterization is not only computationally inefficient, but it may also hinder the explainability of the network, as correlations tend to increase non-linearly when increasing the parameter space.

Many network pruning methods already exist for improving the computational cost at the inference stage [31, 22, 43, 21, 40, 34, 23]. In pruning methods, the model is still trained as usual with over-parameterization in place, but at a second stage, the unimportant connections in the network will be removed based on a certain criterion (see Figure 1).

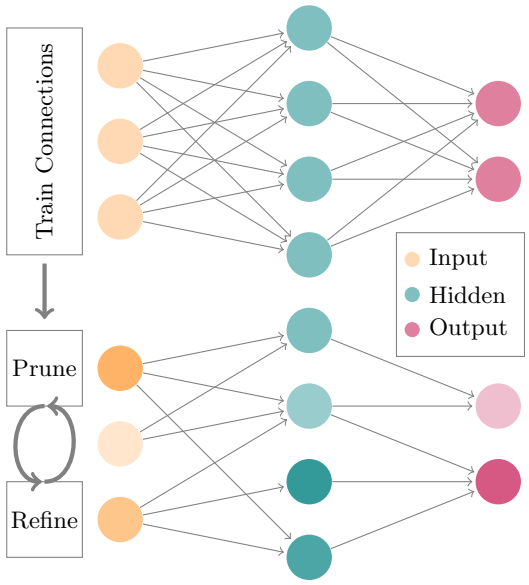


Figure 1. An illustration of a typical pruning pipeline.

After that, the pruned model can be fine-tuned to account for a potential reduction in accuracy [36]. Many adaptations exist that, in essence, try to compress a dense model to a sparse model. Yet this dense-to-sparse training, as performed in, e.g., Single-Shot Network Pruning [32] or Dynamic Network Surgery [18], still requires the initial memory footprint of the dense network. The main benefit of such pruning methods seems to not be given by inheriting important weights, but by the resulting sparse architecture [36, 14, 15]. These methods can thus be seen

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

as a form of topological neural network search. To keep up with ever more challenging problems, simply scaling the existing dense-to-sparse models may turn out not to be enough [47, 26].

Only recently, more effort has been put into developing completely sparse-to-sparse training methods in which the neural network can dynamically alter the topology throughout training while keeping the number of connections at all times bounded [38]. Deep Rewiring [6] looks at the problem theoretically by altering the back-propagation algorithm to incorporate a simultaneous optimization of the connectivity graph. Evolutionary algorithms such as Sparse Evolutionary Training (SET) [39] and RigL [13], on the other hand, take inspiration from biological phenomena. SET has shown to produce scale-free topologies [4, 8] and is based on on-the-fly connection pruning/regrowing while staying sparse throughout the training process.

Further reductions in computational costs can be achieved through feature extraction and feature selection. Dimensionality reduction can be vital in reducing model complexity and can reduce over-fitting. Feature extraction algorithms derive useful information from the data to transform the data onto a new feature space [20]. The goal is to compress the data while keeping the most relevant information. However, the memory requirements may still be similar, given that all features are still being used. Feature selection, on the other hand, will filter the existing features into a subset of features. It has the potential to reduce the model complexity, improve the generalization capability and explainability of the model [19]. Since some sparse training algorithms (e.g. SET [39] and RigL [13]) seem to encode importance to the input nodes, this weight information could potentially be used for feature selection. For unsupervised training of Autoencoders, a truly sparse selection process called *QuickSelection* has shown promising results by reducing the parameter size by at least an order of magnitude while significantly increasing the computational efficiency [3].

Many areas of sparse training and feature selection of DNNs are still unexplored [38]. For supervised learning methods like MLPs it is yet unclear whether the weights of the input neurons will correlate with the feature importance. Moreover, once features are selected, it is still an open question whether such a feature selection is robust to topological changes in the network.

The goal of this research is thus to understand the behaviour of SET-MLPs for supervised feature selection concerning robustness. It ultimately should inform whether the recent trends into sparse training will pay-off as good memory-efficient candidates for going beyond state-of-the-art DNNs. To pursue our goal, we have defined the following research questions as the basis of our research:

Limits *How far can we stress SET-MLP in terms of sparsity and the number of features selected while retaining accuracy?*

Similarity *How similar is the feature selection of different SET-MLPs trained on the same dataset?*

Initialization-dependence *How does SET-MLP respond to bad initializations?*

Generalizability *How generally applicable are SET-MLPs for feature selection?*

In this paper, we study these open questions to start understanding how robust supervised feature selection with SET-MLPs is to topological changes in the network. We make the following specific contributions:

(1) We develop a supervised feature selection method using SET-MLPs.

(2) We create a unique framework designed for sparse training methods to perform a broad evaluation of their robustness characteristics for feature selection.

(3) We repeat some experiments by Liu et al. [35] on Sparse Topology Metrics and improve the original algorithm and implementation. We then analyze the similarity of the network topology obtained by different SET runs that are used for feature selection.

(4) We establish an *inverse problem* in which a sparse-to-sparse network has to find the complete opposite feature importance from what it would select after initialization.

The remainder of this paper is organized as follows: Section 2 discusses related work in sparse-to-sparse training and feature selection by illustrating the general concepts. After that, the specific methodology and experimental setup in use to answer the above research questions is discussed in Section 3 and 4. In Section 5, the results are presented and discussed in Section 6. Finally, the limitations and open questions of the study are discussed, resulting in possible future directions.

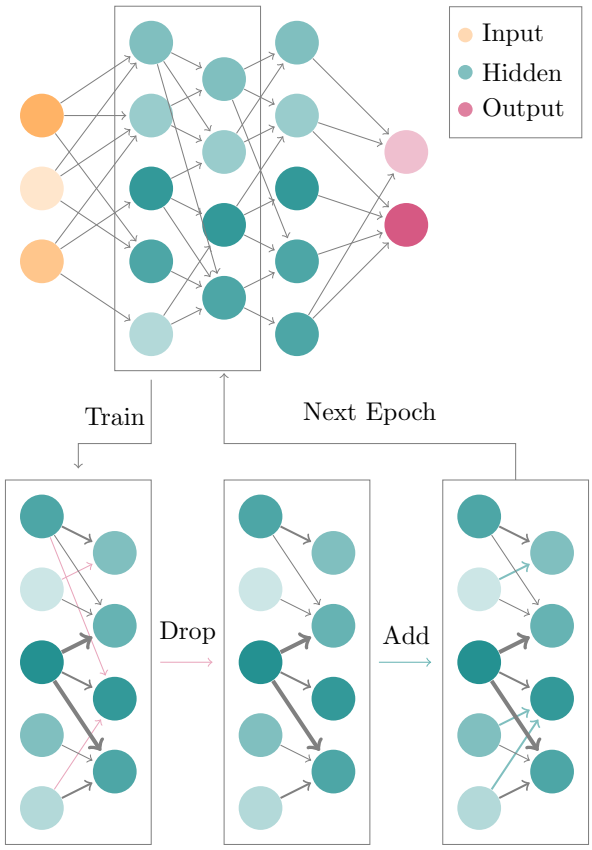


Figure 2. Illustration of the SET algorithm for a MLP.

2. RELATED WORK

In this section, we introduce related work in sparse neural networks, feature selection and topological similarity metrics that are the basis of this study.

While DNNs have been a topic of research for several decades, the benefits of sparse architectures are still being explored. Gale et al. [15] extensively evaluate many fixed pruning and sparsification methods. The experiments indicate that simple pruning techniques can achieve equal

or better performance than complex pruning techniques that may perform more inconsistently. In addition, they observe that it is more difficult to train a fixed learned sparse architecture through pruning than a model that incorporates sparsification in the optimization and training process [15]. The resulting sparse architecture seems to be the crucial aspect. For a traditional example, see Single-Shot Network Pruning (SNIP), where Lee et al. [32] attempt to find a mask to filter an initially dense neural network before training. After pruning, training proceeds as usual with the static, sparse network. An additional limitation to pruning is that the initial memory footprint is still required, which may not be suitable for low resource environments.

Various types of sparse-to-sparse training techniques exist that could be used as the basis of the study [38]. Mocanu et al. have proposed the concept of Sparse Evolutionary Training (SET) in 2018 [39] after initially attempting to use a fixed sparse connectivity pattern in 2016 [37]. SET uses Erdös-Rényi graph initialization to create the initial sparse network. It is a graph where each edge in the same layer has an equal probability to be taken. SET then takes the currently trained sparse network and removes a fraction of the weights with the smallest magnitude. It then randomly creates as many new connections across the network as were removed. Figure 2 visualizes this method. SET is similar to an evolutionary process in that it can produce a sparse, scale-free network topology. As a result, it requires significantly fewer connections and quadratically reduces the number of parameters at no decrease in accuracy [39, 7]. To allow for faster convergence than random evolution, RigL [13] and Top-KAST [27] make use of the gradient information from non-existing connections during the regrow phase. However, the information needed from these non-existent connections may turn out to not be scalable enough. A similar strategy that is based on the momentum of each parameter has also been explored by Dettmers and Zettlemoyer [12]. Given that the robustness of such methods will be analysed, the convergence rate is of secondary importance. Instead, the simplicity of SET creates a transparent platform for further study. In addition, the random regrow strategy makes SET less biased by gradient information which is preferred in a study of robustness.

Sparse training can also be adapted to other applications such as feature selection resulting in lower computational costs for larger data sets. The work by Atashgahi et al. [3] demonstrates that the ideas developed for SET can map over to unsupervised training of Autoencoders to perform unsupervised feature selection. In essence, the weights of the input neurons are taken as a metric for choosing the important features. Whether these methods will translate to supervised training on SET-MLPs is still unclear. To gain further insights into the topological structures of sparse DNNs, Liu et al. [35] propose a Neural Network Sparse Topology Distance (NNSTD) metric. It enables one to compare and visualize different sparse network topologies. NNSTD treats the neural network as a neural graph such that it can use the well-known Graph Edit Distance (GED) [42]. It considers the minimum distance needed to transform a graph g_1 into another graph g_2:

$$GED(g_1, g_2) = \min_{\forall p \in \mathbb{P}(g_1, g_2)} c(p), \tag{1}$$

where p is a sequence of transformations in the set of all possible transformations $\mathbb{P}$ that has a total cost of $c(p)$.

The idea is then to take two layers at a time from both networks, and for each combination of neurons, the delta between the connections is computed in terms of the Normalized Edit Distance (NED) given by:

$$NED(g_1, g_2) = \frac{|g_1 \setminus g_2 \cup g_2 \setminus g_1|}{|g_1 \cup g_2|}, \tag{2}$$

where g_1, g_2 are two layers coming from two networks.

3. PROPOSED METHODS

To address the research questions regarding the robustness of supervised feature selection using sparse neural networks, this section details the proposed methods.

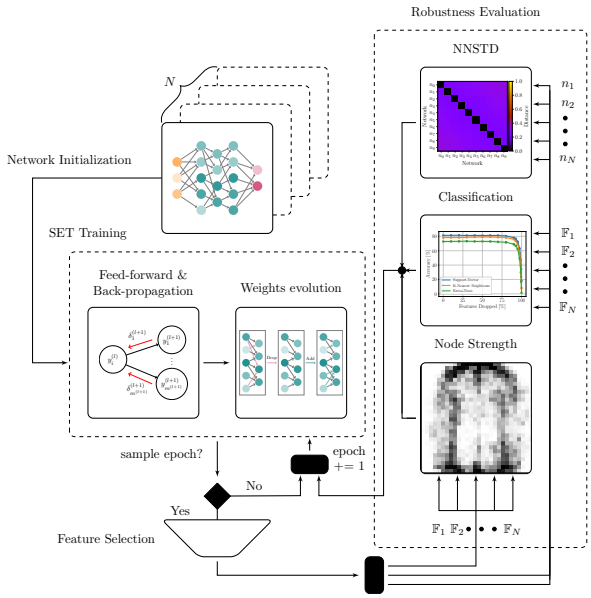


Figure 3. Illustration of Evaluation Framework (see Section 3.1 for a description), where N is the number of runs, n the n^{th} run and $\mathbb{F}_i$ the i^{th} selected feature set.

3.1 Evaluation Framework

The evaluation framework follows closely the proposed research questions of Section 1 (see Figure 3). In general, various SET-MLPs are trained on the datasets and used for feature selection as discussed in Section 3.2. The selected features are then used to perform classification with various standard classifiers (see Section 4), and the accuracy on these tasks will be recorded. We repeat the general setup for various sparsity levels of the SET-MLP. We not only apply feature selection at the end of training SET, but perform the same evaluation after several sample epochs. To visualize the feature selection, several images are created which indicate the prevalence of the features over multiple runs (see Section 3.3). Finally, the Network Topology Distance is used to judge the similarity of the underlying SET networks (see Section 2 and 3.3).

3.2 Feature Selection

Various feature selection mechanisms can be thought of that use the weight and connectivity information of SET-MLP. Since the focus of this research is not on finding the best possible feature selection approach, only a simple, straightforward method is introduced and used for feature selection. Concretely, first a SET-MLP is trained as defined in [39] and visualized in Figure 2. Afterwards, the first weight layer representing the input neurons will be

used to determine the node strengths. This was also already done by Atashgahi et al. [3] for unsupervised feature selection and follows the typical graph theory conception of node strength. A node strength s is determined by taking the sum of the magnitude of all its outgoing connections:

$$s_i = \sum_{j=1}^{N^1} |w_{i,j}^1|, \tag{3}$$

where N^1 represents the number of neurons in the first hidden layer of the network. Similarly, $w_{i,j}^1$ is the weight associated with input neuron i and the j^{th} neuron of the first hidden layer.

This strength can now be used to rank the features such that finally, only the most important percentage of features is selected:

$$\mathbb{F} = \{f_i | i \in \mathbb{Z}, 0 \leq i \leq N^0, s_i \geq \mathbb{S}_k\}; k = \lfloor N^0 \lambda \rfloor, \tag{4}$$

where $\mathbb{F}$ is the resulting set of selected features, f_i the i^{th} feature, $\mathbb{S}_k$ the k^{th} element of the sorted set (descending) of node strengths, N^0 the number of input neurons, and λ the percentage of features to be selected.

3.3 Network Similarity Metrics

Quantifying the similarity of different SET-MLP topologies can be achieved through various means.

Images.

The first method in use visualizes the features as an image where each pixel represents a different feature. It is especially useful in image classification tasks to get an intuitive understanding of the feature selection. It can be used to show the evolution of the feature selection after various training epochs of the SET-MLP. In addition, different intensity levels of the colour can provide insight into spatially locating the essential features.

Sparse Topology Distance.

To better quantify the similarity of different topologies, the method introduced by Liu et al. [35] is also used here and explained in Section 2. Given the complexity of this algorithm, adding more runs into the comparison will drastically increase the computation time. Therefore, we improved the computational cost of the inner loop of the official implementation through typical optimization strategies (use of Numba [29], a JIT backend to Python)[1]. Also, by identifying that the elements in g_1 and g_2 are all unique, the computation of the Normalized Edit Distance (NED) can further be simplified to:

$$d(g_1, g_2) = \frac{|g_1| + |g_2| - 2|g_1 \cap g_2|}{|g_1 \cup g_2|}, \tag{5}$$

where d represents the NED and g_1, g_2 two layers coming from two networks.

4. EXPERIMENTAL SETUP

To assess the robustness of feature selection using SET-MLP, several experiments have been conducted to give insight into the research questions in a step-wise fashion. Note that we explicitly do not look at the raw accuracy, but limit this study to the specific robustness characteristics. To begin with, the first experiment tests the classification accuracy for various classifiers and varying numbers of features selected. The feature selection is performed using SET-MLP as described in Section 3.2. The classifiers that are taken into consideration are support-vector [46], k-nearest neighbours [1] and extra-trees classifier [16]. The general characteristics of a fixed sparsity level for SET-MLP are first explored in detail. Then, the same experiment is performed for different sparsity levels of SET. For all experiments, the number of features selected ranges from 0% to 100% in 10% increments. It allows to explore the scaling behaviour of the feature selection and may highlight the potential critical percentage of features needed to still perform the classification task with reasonable accuracy.

4.1 Initialization Dependence

Much previous work already points out the sensitivity of various neural networks to their initialization [45]. This is also tested for SET-MLP by using different random seeds for each run. In combination with the other aspects of the evaluation, this should provide a solid ground to explore whether different seeds create different resulting networks and feature selections. Furthermore, a *worst-case* scenario is performed, in which the SET-MLP is first trained as normal. After that, the process is repeated, however, the most important features that have been found are disabled completely during initialization by setting all the corresponding connections to 0. The rest is initialized randomly as usual. Note that the sparsity level is nonetheless the same, resulting in more unimportant weights. This process can loosely be thought of as the *inverse problem* in which the network has to find the complete opposite from what it received as input. The performance of this task can then be evaluated by comparing the feature selection of the initial run with that of the modified one. The specific configurations of these experiments given a dataset are provided in the next sections.

Table 1. Dataset properties

Name	Features	Type	Samples	Classes
F-MNIST	784	Image	70000	10
Lung	3312	Microarray	203	5

4.2 Datasets

It has been shown in the past that specific attributes of a dataset can significantly alter the performance of a model [33, 41]. Therefore, it is vital to perform the above-mentioned tests on a variety of datasets with different characteristics. The default dataset, with which the most detailed analysis is performed, is Fashion-MNIST [49]. It poses an image classification problem and is particularly useful in getting an intuitive understanding of the behaviour of an algorithm. Since it has 28×28 pixels per image, this dataset has 784 features and 10 possible output classes. From the biological domain, Lung from the scikit-feature feature selection repository [33] is used as another continuous, multi-class dataset with 3312 features and 5 output classes. Table 1 provides an overview of the dataset characteristics.

4.3 Hyperparameters

SET has various hyperparameters that can alter the performance of the model. Throughout the test process, all

[1]The code is available at: `https://git.snt.utwente.nl/s2106019/sparse_topology_distance`

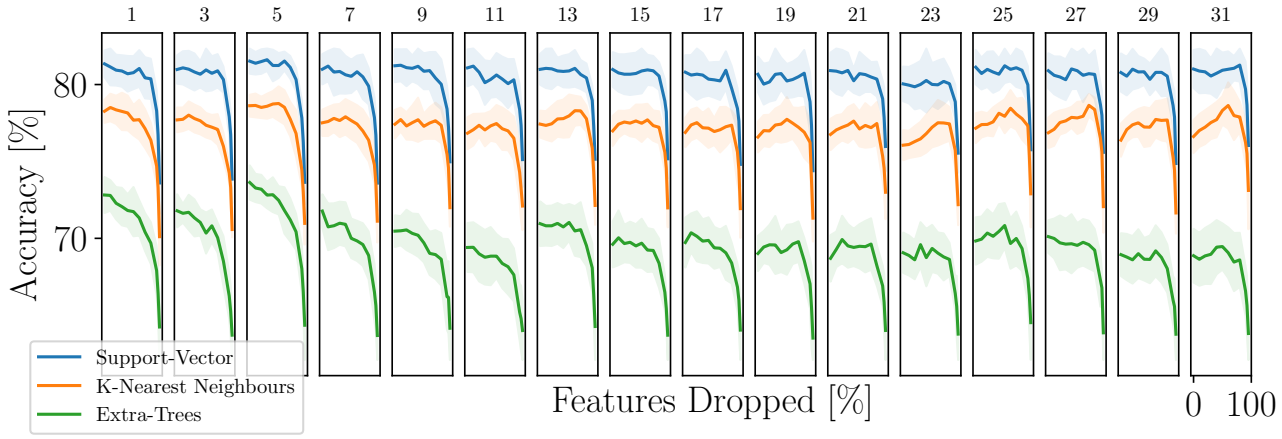


Figure 4. In each plot, the classification accuracy of support-vector, k-nearest neighbour and extra-trees classifier is plotted against the percentage of features dropped ranging from 0% to 100%. *Features Dropped* indicates the percentage of features that were removed by the feature selection. Every plot represents a different sparsity level of SET's underlying network ranging from using $\epsilon = 1, 3, ..., 31$ (see top label).

these parameters are held constant unless explicitly stated otherwise. The fraction of weights removed at each epoch is $\zeta = 0.3$. So, 30% of the weights are removed and replaced at each epoch using the process illustrated in Figure 2. The sparsity of the network is based on the variable ϵ, which determines how many edges are selected during initialization. The sparsity level is by default at $\epsilon = 13$. It, for instance, roughly equates to a 2% density level $(1 - sparsity)$ in the first hidden layer of Fashion-MNIST. Besides, Nesterov momentum is used during optimization with $\alpha = 0.9$, and the batch size is 40 and 2 for Fashion-MNIST and Lung, respectively. All data is normalized to mean 0 and unit variance. We train SET for 400 epochs using a learning rate of $\eta = 0.05$ and a weight-decay of 2e−4. No dropout is used during this process.

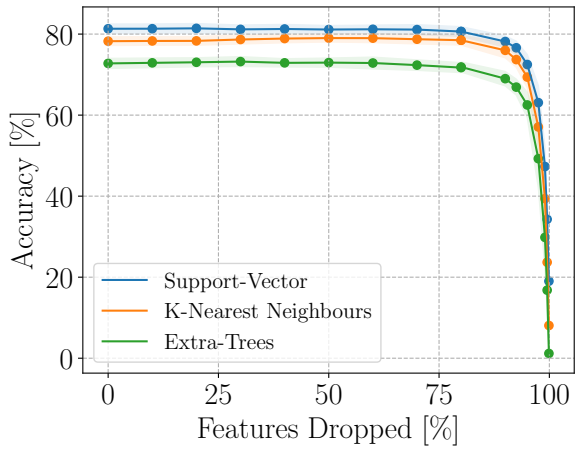


Figure 5. Classification accuracy of support-vector (green), k-nearest neighbours (red) and extra-trees (blue) classifier for different feature selections. The feature selection is performed using the proposed method (see Section 3.2). 32 runs, Fashion-MNIST, $\epsilon = 13$.

4.4 Implementation

The implementation of SET[2] is written in Python and uses purely sparse datastructures for working with the weights. For this, SciPy's sparse matrices are used [48]. In particular, the second listed implementation is the foundation of this research and uses Numba [29] for the performance-critical sections. We have adapted the code slightly to fit our needs and provide feature selection and various plotting functions[3]. For the datasets Fashion-MNIST and Lung, 3 hidden layers with each 3000 neurons are used for SET. Besides, we use $\frac{2}{3}$ of the data as training data and the rest for validation. The split size is the same for both training of SET and the classifiers. In each run, a different random split is taken and SET will have a different initialization.

4.5 Evaluation Environment

The experiments were performed on a quad-core CPU, namely an Intel(R) Core(TM) i7-8550U CPU @1.8 GHz with a maximum clock speed of 4 GHz run on Linux. The implementation can use all the available cores, however, no GPUs were used during this process given the limited hardware support for sparse methods.

5. RESULTS

This section presents the findings of the experiments mentioned in Section 4.

Baseline Accuracy at fixed sparsity levels.

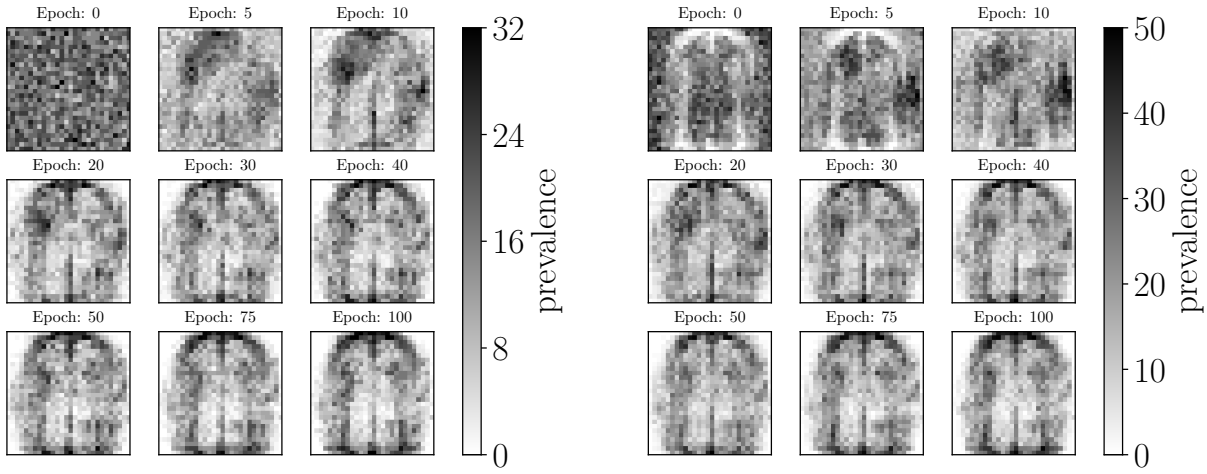
First, we explore how many features can be dropped for the classification task before seeing noticeable accuracy degradation. When looking at Figure 5, we can see that the accuracy is unaltered for up to 80% dropped features. The accuracy of all classifiers reduce by $3 - 5\%$ at 10% of remaining features and drops significantly afterwards. This behaviour is reasonable since, at some point, there exist too few features to make meaningful predictions. More interesting is the fact that the variance of the accuracy remains small for the 32 runs. These accuracies form a baseline for the following experiment.

Accuracy at different sparsity levels.

Next, we explore how the accuracy of the different classifiers changes if SET is performed at varying sparsity levels.

[2]The official SET repository from which the second listed implementation is used can be found on GitHub.

[3]The code of this paper is available at: `git.snt.utwente.nl/s2106019/robustness_set`



(a) Erdös-Rényi graph initialization. **(b) *Worst-Case* initialization.**

Figure 6. Node strength visualization of SET-MLPs underlying network using (a) Erdös-Rényi and (b) *worst-case* initialization (see Section 4.1). 70% of features dropped, Dataset is Fashion-MNIST.

We vary ϵ from 1 to 31 and perform the same evaluation as before. Looking at Figure 4, we mostly see that the performance of the classifiers remains stable but drops faster for very low densities. The deviations from low to high sparsity are more pronounced for smaller ϵ. This is most notable in the k-nearest neighbours classifier, where the accuracy stabilizes for $\epsilon = 13$ even at 90% dropped features.

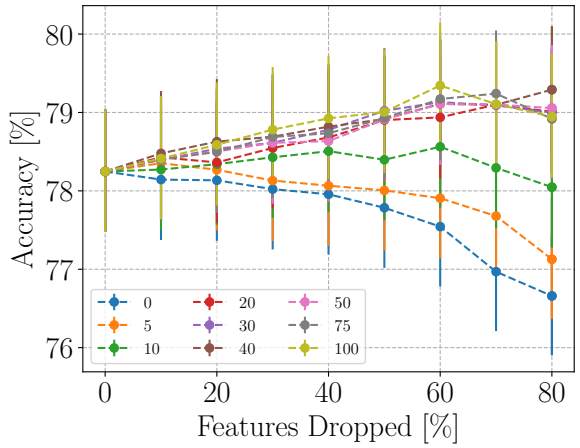


Figure 7. K-Nearest Neighbours classifier accuracy for different feature selections. The legend shows the colours of the different training epochs of SET-MLP before the feature selection was performed. Based on 32 runs, Fashion-MNIST.

Accuracy at various stages of SET.

To further explore the true impact of SET feature selection on the accuracy of classifiers like k-nearest neighbours, we let SET train for $0, 5, 10, 20, 30, 40, 50, 75$ and 100 epochs before performing feature selection (see Figure 7). We can identify that after 20 epochs, a boundary occurs whereby the accuracy remains very close together and in the error margin of the experiment. More importantly, the plot shows that for an increased number of dropped features, the difference between training and no training - thus random feature selection - becomes more pronounced. For the case of the k-nearest neighbours classifier, the accuracy increases from 77% to 79% at 20% remaining features. Figure 11a and 11b contain similar results for the support-vector and extra-trees classifier, respectively.

Feature selection similarity.

Given that the accuracy remains stable for the various classifiers, it remains to see whether the actual feature selection by SET remains stable as well. For this, the 28×28 pixels representing the features of Fashion-MNIST are given a prevalence score at various stages of the SET training phase. The prevalence score counts in how many runs the feature selection selected the feature (see Figure 6a). The feature selection is performed after training the SET-MLP for the specified number of epochs and removes 70% of the features. Initially, the distribution is random due to the Erdös-Rényi graph initialization. After 20 epochs, the feature pixels located at the borders are removed, which is reasonable given that most pictures are black at the borders in Fashion-MNIST. However, also the center is given less importance, indicating that this method selects features on the contours of most clothing.

Feature selection inverse problem.

Another robustness criterion mentioned as a research question was the initialization dependence of SET. That is, does SET require specific initialization - which in this case is Erdös-Rényi graph - or is it robust to varying initial networks? We establish a worst-case scenario as described in Section 4.1 by first training SET to find the important features to then explicitly remove those features in the initialization of the actual run. It only alters the first hidden layer and keeps other layers Erdös-Rényi initialized. An *inverse* problem is created, which should ideally result in the inverse image at the final epoch (assuming that the initial features were indeed the important ones). Figure 6b presents this process in which SET successfully recovers its initial feature importance.

Topological similarity of SET.

It remains to investigate whether the different runs of SET produce different networks or if very similar network structures can emerge. For this investigation the Net-

work Topology Distance (NNSTD) is used as discussed in Section 2 and 3. Based on the NNSTD, the SET networks that were used for feature selection (after epoch 400) are all similarly distant from each other, hovering around 0.87 ± 0.01. So, even though the feature selection remains almost the same, the actual network topology does not. It indicates that many networks exist that perform well. The plot on the right side of Figure 8 shows the evolution of a single network. The results seem to be independent of the initialization strategy. We can observe that the similarity is largest and spreads outwards from the diagonal line. This is in line with expectation given that one expects, e.g., epoch 40 to be more similar to epoch 30 and 50 than epoch 300. The NNSTD suggests that the network keeps on changing even after 300 epochs. Interestingly, the initial epochs are much closer together. Overall, the NNSTD remains above 0.5 in all cases.

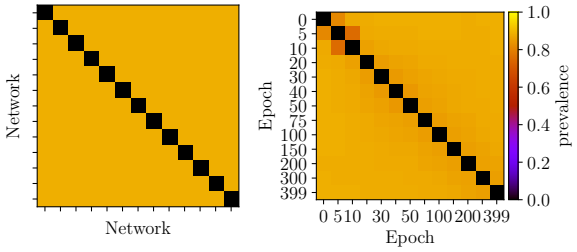


Figure 8. Network Topology Distance for Fashion-MNIST. On the left, networks from different runs are compared for topological similarity. The right figure shows one specific SET network at different stages of training. The missing label values are always in between the two neighbouring labels.

Generalizability.

To start exploring the generalizability of this method, we also investigate the performance on Lung [33]. For Lung, the general accuracy characteristics remain the same (see Figure 9), however, higher variance in the accuracy can be observed. This is to be expected given that Lung contains only a few hundred samples. In Figure 10, we also observe that training SET-MLP becomes more important once more than 50% of the features are dropped. However, overall, the higher variance overshadows any significant gains. For the extra-trees classifier, these findings are more pronounced. Additional results for the extra-trees and support-vector classifier are given in Figure 11c and 11d, respectively. The Madelon dataset has also been used and shows some of the limitations of this method when SET-MLP does not converge completely in the given time and thus results in poor feature selection[4]. Madelon is an artificially created dataset consisting of many noisy, redundant features which makes it more difficult for SET to gain any useful information most of the time. Given the random evolutionary nature of SET, the convergence may thus take a long time. Further analysis is beyond the scope of this research, given that feature selection should be reasonably fast to be useful.

6. DISCUSSION

In this paper, we provided several perspectives into answering whether feature selection using SET-MLPs is robust. Connecting the initial findings of Fashion-MNIST, we observe a qualitative difference after training SET for

[4]For results on the Madelon dataset see the Addendum

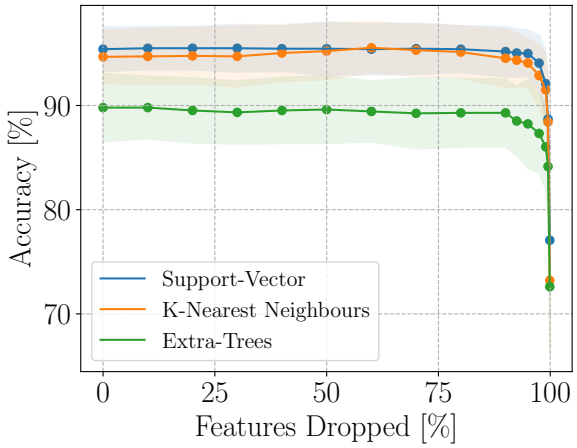


Figure 9. Classification accuracy of support-vector (green), k-nearest (red) and extra-trees (blue) classifier for different feature selections. The feature selection is performed using the proposed method (see Section 3.2). 32 runs, Lung, $\epsilon = 13$.

20 epochs. Here the accuracy increases (as seen in Figure 7) since SET removes the unnecessary boundary features present in Fashion-MNIST pictures (see Figure 6a). It is likely not a coincidence, as many runs come to the same results even given a bad initialization (see Figure 6b). The convergence of SET is fast in these cases and thus establishes itself as a practical method for feature selection. The similarity of the networks turns out to be different given the NNSTD metric. This is in line with the findings of Liu et al. [35]. Overall, the observed distance is slightly smaller compared to the existing results. This may be explained by the fact that we used 3000 hidden neurons per layer compared to 784 hidden neurons. In general, the Network Topology Distance metric is very sensitive to changes in the network structure, resulting in almost everything getting the same large distance score.

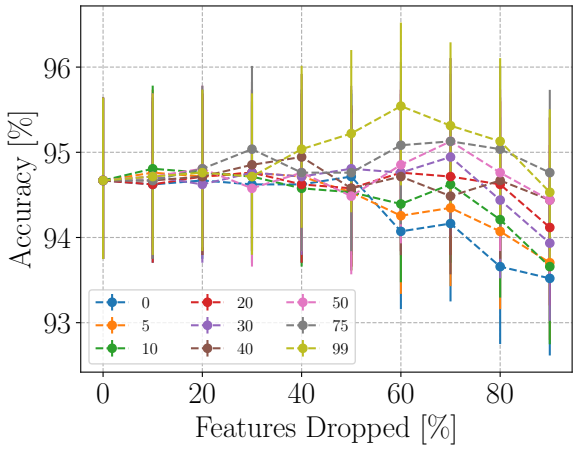


Figure 10. K-Nearest Neighbours classifier accuracy for different feature selections. *Features Dropped* indicates the percentage of features that were removed by the feature selection. The legend shows the colours of the different training epochs of SET-MLP before the feature selection was performed. Based on 32 runs, Lung.

Liu et al. discuss very minor changes in the network, which can be picked up using this metric. However, for a more coarse-grained analysis, NNSTD is of limited use. Other

similarity metrics would have to be created given that, to our knowledge, no other methods exist in the literature.

These findings also create a connection to the *Lottery Ticket Hypothesis* [14]. It states that there exist subnetworks of randomly initialized, dense neural networks which can achieve the same accuracy as the initial network after training at most the same number of iterations. SET seems to find such substructures more systematically and is one reason why feature selection is possible with it.

7. CONCLUSION

In this work, we introduced a procedure to evaluate the robustness of SET-MLPs for use in supervised feature selection. Notwithstanding the limitations mentioned in Section 8, we demonstrated that SET-MLPs indeed produce similar feature selections while the underlying networks may differ significantly. Even in worst-case initialization scenarios, SET can select meaningful features, providing a solid basis for a robust feature selection mechanism. After several epochs, the benefits of SET-MLP feature selection are already present, making it a viable feature selection strategy beyond traditional methods. The generalizability of this method has been experimentally tested on various datasets, however, it remains to be seen whether it holds up for larger-scale datasets with many more output classes. Ultimately, this research provides a hopeful future for sparse training and feature selection, which has the potential to drastically decrease the memory footprint and computational cost of new and existing classification methods.

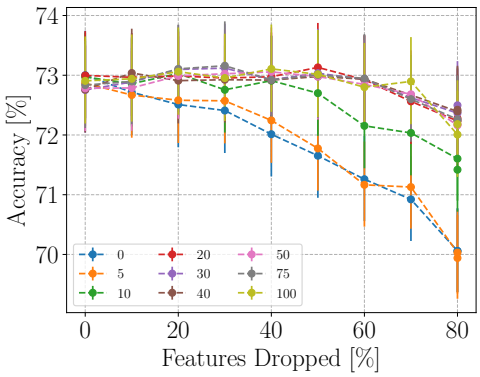
8. LIMITATIONS AND FUTURE WORK

This paper investigates the feature selection capabilities of SET using a rather simple selection mechanism (see Section 3.2). It cannot be excluded that better feature selection strategies exist. An elaborate study of various techniques is warranted, and future work may look into possible new yet unknown approaches. The qualitative aspect of the results is, however, likely to not change and is the reason for choosing this simple mechanism. As with any experimental study, one should analyse the behaviour on many more datasets. Given the few output classes of the chosen datasets, it will be interesting to see whether the results hold up for larger, more complex datasets. The Madelon dataset already gives some indications of the limitations of this method. If SET-MLP cannot quickly enough converge, it will result in bad feature selections.

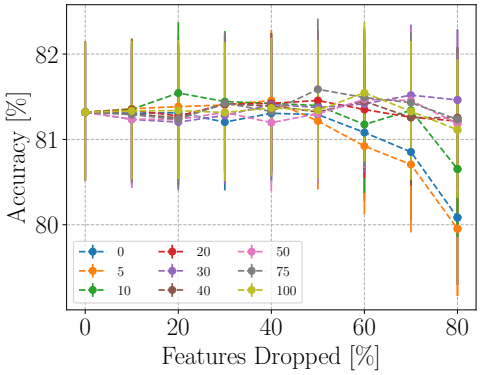
In terms of analysing the robustness of this method, many other perspectives can be thought of. It may turn out that on larger datasets, the feature selection ignores certain, uncommon output classes. Hooker et al. [25] already looked into whether compressed models can forget. They form an analysis on dis-aggregated measures of model performance, which could also be used in similar studies on feature selection using sparse-to-sparse training. It is also interesting to see whether the robustness characteristics of this approach can extend to robustness to adversarial attacks on the dataset. It has been explored for image classification by Goodfellow et al. [17], Carlini and Wagner [9] and Bastani et al. [5] among others, but similar studies are lacking for feature selection strategies.

9. ACKNOWLEDGMENTS

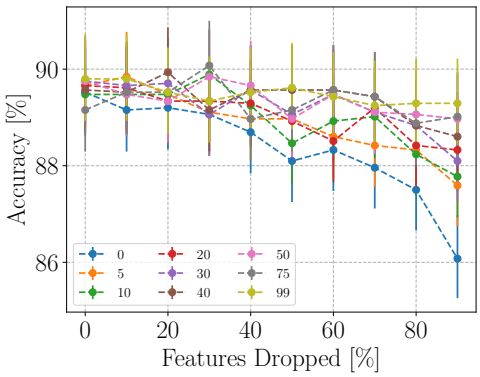
I thank my supervisors, Decebal Mocanu and Zarah Atashgahi for continuous and insightful feedback to both writing and executing this thesis.



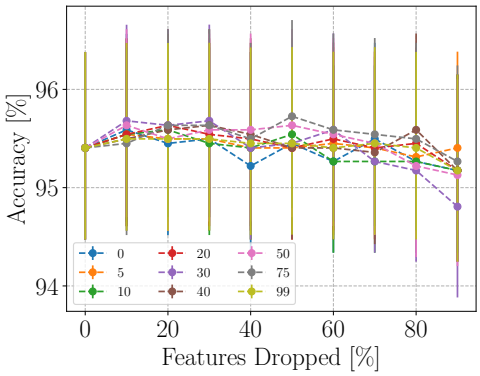
(a) Fashion-MNIST: Extra-Trees classifier.



(b) Fashion-MNIST: Support-Vector classifier.



(c) Lung: Extra-Trees classifier.



(d) Lung: Support-Vector classifier.

Figure 11. Additional classification results. Mean classifier accuracy for different feature selections based on 32 runs. *Features Dropped* indicates the percentage of features that were removed by the feature selection. The legend shows the colours for different training epochs of SET-MLP.

References

[1] Naomi S. Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *American Statistician* 46, 3 (1992), 175–185. https://doi.org/10.1080/00031305.1992.10475879

[2] Dario Amodei, Danny Hernandez, Girish Sastry, Jack Clark, Greg Brockman, and Ilya Sutskever. 2018. AI and Compute. https://openai.com/blog/ai-and-compute/

[3] Zahra Atashgahi, Ghada Sokar, Tim van der Lee, Elena Mocanu, Decebal C. Mocanu, Raymond Veldhuis, and Mykola Pechenizkiy. 2020. Quick and Robust Feature Selection: the Strength of Energy-efficient Sparse Training for Autoencoders. *arXiv* (12 2020). http://arxiv.org/abs/2012.00560

[4] Albert L. Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *Science* 286, 5439 (10 1999), 509–512. https://doi.org/10.1126/science.286.5439.509

[5] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. 2016. Measuring Neural Net Robustness with Constraints. *Advances in Neural Information Processing Systems* (5 2016), 2621–2629. http://arxiv.org/abs/1605.07262

[6] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. 2017. Deep Rewiring: Training very sparse deep networks. *arXiv* (11 2017). http://arxiv.org/abs/1711.05136

[7] David D. Bourgin, Joshua C. Peterson, Daniel Reichman, Stuart J. Russell, and Thomas L. Griffiths. 2019. Cognitive Model Priors for Predicting Human Decisions. In *Proceedings of the 36th International Conference on Machine Learning*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Ed.). PMLR, 5133–5141. http://proceedings.mlr.press/v97/peterson19a/peterson19a.pdf

[8] Ed Bullmore and Olaf Sporns. 2009. Complex brain networks: Graph theoretical analysis of structural and functional systems. , 186–198 pages. https://doi.org/10.1038/nrn2575

[9] Nicholas Carlini and David Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *Proceedings - IEEE Symposium on Security and Privacy*. Institute of Electrical and Electronics Engineers Inc., 39–57. https://doi.org/10.1109/SP.2017.49

[10] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. 2013. Predicting Parameters in Deep Learning. *Advances in Neural Information Processing Systems* (6 2013). http://arxiv.org/abs/1306.0543

[11] Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. 2014. Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation. *Advances in Neural Information Processing Systems* 2, January (4 2014), 1269–1277. http://arxiv.org/abs/1404.0736

[12] Tim Dettmers and Luke Zettlemoyer. 2019. Sparse networks from scratch: Faster training without losing performance. *arXiv* (7 2019). https://arxiv.org/abs/1907.04840

[13] Utku Evci, Trevor Gale, Jacob Menick, Pablo S. Castro, and Erich Elsen. 2020. Rigging the Lottery: Making All Tickets Winners. In *Proceedings of the 37th International Conference on Machine Learning*, Hal Daumé III and Aarti Singh (Eds.). 2943–2952. http://proceedings.mlr.press/v119/evci20a/evci20a.pdf

[14] Jonathan Frankle and Michael Carbin. 2018. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *arXiv* (3 2018). http://arxiv.org/abs/1803.03635

[15] Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The State of Sparsity in Deep Neural Networks. *arXiv* (2 2019). http://arxiv.org/abs/1902.09574

[16] Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine Learning* 63, 1 (4 2006), 3–42. https://doi.org/10.1007/s10994-006-6226-1

[17] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR. https://arxiv.org/abs/1412.6572

[18] Yiwen Guo, Anbang Yao, and Yurong Chen. 2016. Dynamic Network Surgery for Efficient DNNs. *Advances in Neural Information Processing Systems* (8 2016), 1387–1395. http://arxiv.org/abs/1608.04493

[19] Isabelle Guyon and Andre Elisseeff. 2003. An Introduction to Variable and Feature Selection André Elisseeff. *Journal of Machine Learning Research* 3 (2003), 1157–1182. https://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf?ref=driverlayer.com/web

[20] Isabelle Guyon and André Elisseeff. 2006. An Introduction to Feature Extraction. In *Feature Extraction*. Vol. 207. Springer Berlin Heidelberg, Berlin, Heidelberg, 1–25. https://doi.org/10.1007/978-3-540-35488-8_1

[21] Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both Weights and Connections for Efficient Neural Networks. *Advances in Neural Information Processing Systems* 2015-Janua (6 2015), 1135–1143. http://arxiv.org/abs/1506.02626

[22] Babak Hassibi and David G. Stork. 1992. Second order derivatives for network pruning: Optimal Brain Surgeon. In *Proceedings of the 5th International Conference on Neural Information Processing Systems*. Morgan Kaufmann, 164–171.

[23] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. 2018. AMC: AutoML for Model Compression and Acceleration on Mobile Devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 784–800. https://arxiv.org/abs/1802.03494v4

[24] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Mostofa A. Patwary, Yang Yang, and Yanqi Zhou.

2017. Deep Learning Scaling is Predictable, Empirically. *arXiv* (12 2017). http://arxiv.org/abs/1712.00409

[25] Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. 2019. What Do Compressed Deep Neural Networks Forget? *arXiv preprint arXiv:1911.05248* (11 2019). http://arxiv.org/abs/1911.05248

[26] Mark Horowitz. 2014. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 10–14. https://doi.org/10.1109/ISSCC.2014.6757323

[27] Siddhant M Jayakumar, Razvan Pascanu, Jack W. Rae, Simon Osindero, Deepmind Erich, and Elsen Deepmind. 2021. Top-KAST: Top-K Always Sparse Training. *Advances in Neural Information Processing Systems* (2021). https://arxiv.org/abs/2106.03517

[28] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-Luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara V. Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, Richard C. Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon Mackean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe H. Yoon. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ACM, New York, NY, USA. https://doi.org/10.1145/3079856.3080246

[29] Siu K. Lam, Antoine Pitrou, and Stanley Seibert. 2015. Numba: A LLVM-based Python JIT Compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC - LLVM '15*. ACM Press, New York, New York, USA, 1–6. https://doi.org/10.1145/2833157.2833162

[30] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. , 436–444 pages. https://doi.org/10.1038/nature14539

[31] Yann LeCun, John S. Denker, and Sara A. Solla. 1990. Optimal Brain Damage. In *Advances in Neural Information Processing Systems (NIPS 1989)* (2 ed.). Morgan Kaufmann, Denver, CO, 598–605. https://papers.nips.cc/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf

[32] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. 2018. SNIP: Single-shot Network Pruning based on Connection Sensitivity. *arXiv* (10 2018). http://arxiv.org/abs/1810.02340

[33] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Huan Liu, and Jiliang Tang. 2017. Feature Selection: A Data Perspective. *Comput. Surveys* 50, 6 (2017), 1–45. https://doi.org/10.1145/3136625

[34] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. 2017. Runtime Neural Pruning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2178–2188. https://proceedings.neurips.cc/paper/2017/file/a51fb975227d6640e4fe47854476d133-Paper.pdf

[35] Shiwei Liu, Tim Van der Lee, Anil Yaman, Zahra Atashgahi, Davide Ferraro, Ghada Sokar, Mykola Pechenizkiy, and Decebal C. Mocanu. 2021. Topological Insights into Sparse Neural Networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 12459 LNAI. Springer Science and Business Media Deutschland GmbH, 279–294. https://doi.org/10.1007/978-3-030-67664-3_17

[36] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2018. Rethinking the Value of Network Pruning. https://arxiv.org/abs/1810.05270v2

[37] Decebal C. Mocanu, Elena Mocanu, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. 2016. A topological insight into restricted Boltzmann machines. *Machine Learning* 104, 2-3 (9 2016), 243–270. https://doi.org/10.1007/s10994-016-5570-z

[38] Decebal C. Mocanu, Elena Mocanu, Tiago Pinto, Selima Curci, Phuong H. Nguyen, Madeleine Gibescu, Damien Ernst, and Zita A. Vale. 2021. Sparse Training Theory for Scalable and Efficient Agents. *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)* (3 2021). http://arxiv.org/abs/2103.01636

[39] Decebal C. Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. 2018. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications* 9, 1 (12 2018), 1–12. https://doi.org/10.1038/s41467-018-04316-3

[40] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2016. Pruning Convolutional Neural Networks for Resource Efficient Inference. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* (11 2016). http://arxiv.org/abs/1611.06440

[41] Dijana Oreski, Stjepan Oreski, and Bozidar Klicek. 2017. Effects of dataset characteristics on the performance of feature selection techniques. *Applied Soft Computing Journal* 52 (3 2017), 109–119. https://doi.org/10.1016/j.asoc.2016.12.023

[42] Alberto Sanfeliu, Alberto Sanfeliu, and King S. Fu. 1983. A Distance Measure Between Attributed Relational Graphs for Pattern Recognition. *IEEE Transactions on Systems, Man and Cybernetics* SMC-13, 3 (1983), 353–362. https://doi.org/10.1109/TSMC.1983.6313167

[43] Nikko Ström. 1997. Sparse connection and pruning in large dynamic artificial neural networks. *Fifth European Conference on Speech Communication and Technology* (1997). https://www.isca-speech.org/archive/archive_papers/eurospeech_1997/e97_2807.pdf

[44] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference* (6 2019), 3645–3650. http://arxiv.org/abs/1906.02243

[45] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*. PMLR, 1139–1147. http://proceedings.mlr.press/v28/sutskever13.html

[46] Johan A. K. Suykens and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural Processing Letters* 9, 3 (1999), 293–300. https://doi.org/10.1023/A:1018628609742

[47] Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. 2020. The Computational Limits of Deep Learning. *arXiv* (7 2020). http://arxiv.org/abs/2007.05558

[48] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, Ilhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, Aditya Vijaykumar, Alessandro Pietro Bardelli, Alex Rothberg, Andreas Hilboll, Andreas Kloeckner, Anthony Scopatz, Antony Lee, Ariel Rokem, C. Nathan Woods, Chad Fulton, Charles Masson, Christian Häggström, Clark Fitzgerald, David A. Nicholson, David R. Hagen, Dmitrii V. Pasechnik, Emanuele Olivetti, Eric Martin, Eric Wieser, Fabrice Silva, Felix Lenders, Florian Wilhelm, G. Young, Gavin A. Price, Gert Ludwig Ingold, Gregory E. Allen, Gregory R. Lee, Hervé Audren, Irvin Probst, Jörg P. Dietrich, Jacob Silterra, James T. Webber, Janko Slavič, Joel Nothman, Johannes Buchner, Johannes Kulick, Johannes L. Schönberger, José V. de Miranda Cardoso, Joscha Reimer, Joseph Harrington, Juan L. C. Rodríguez, Juan Nunez-Iglesias, Justin Kuczynski, Kevin Tritz, Martin Thoma, Matthew Newville, Matthias Kümmerer, Maximilian Bolingbroke, Michael Tartre, Mikhail Pak, Nathaniel J. Smith, Nikolai Nowaczyk, Nikolay Shebanov, Oleksandr Pavlyk, Per A. Brodtkorb, Perry Lee, Robert T. McGibbon, Roman Feldbauer, Sam Lewis, Sam Tygier, Scott Sievert, Sebastiano Vigna, Stefan Peterson, Surhud More, Tadeusz Pudlik, Takuya Oshima, Thomas J. Pingel, Thomas P. Robitaille, Thomas Spura, Thouis R. Jones, Tim Cera, Tim Leslie, Tiziano Zito, Tom Krauss, Utkarsh Upadhyay, Yaroslav O. Halchenko, and Yoshiki Vázquez-Baeza. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods* 17, 3 (3 2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[49] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* (8 2017). http://arxiv.org/abs/1708.07747

[50] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. Understanding deep learning requires rethinking generalization. *Commun. ACM* 64, 3 (11 2016), 107–115. http://arxiv.org/abs/1611.03530