Soft Pruning and Latent Space Dimensionality Reduction

Richard Menzies
Department of Computer Science
University of Glasgow
Glasgow, United Kingdom
r.menzies.1@research.gla.ac.uk

Paul Siebert
Department of Computer Science
University of Glasgow
Glasgow, United Kingdom
paul.siebert@glasgow.ac.uk

***Abstract*—Neural Network pruning is commonly used to reduce the size of a neural network, reducing the memory footprint, while maintaining an acceptable loss. However, currently the only approach explored for removing a parameter from a neural network is to remove the parameter suddenly, irrespective of the pruning method, be it one-shot, iterative or sparsity-induced. We hypothesize that this sudden removal will cause the loss of the information contained within the removed parameters, information which could be useful when retraining the neural network after pruning. To resolve this, we propose *Soft Pruning*, a method of slowly decaying parameters out of a neural network. We compare this to one-shot pruning on the vision-based tasks of classification, autoencoding, and latent space dimensionality reduction. In every experiment, Soft Pruning is able to match or outperform one-shot pruning; in classification, Soft Pruning enables pruning to significantly greater extents than one-shot pruning, retaining over 60% accuracy where one-shot pruning becomes equivalent to random guessing. In autoencoding, Soft Pruning is able to achieve up to 17% lower loss after pruning. Finally, applied to latent space dimensionality reduction, Soft Pruning is shown to achieve more than 60% lower loss compared to one-shot pruning.**

***Index Terms*—neural network pruning, latent space optimisation**

I. Introduction

Neural Network pruning originated as a method to reduce the memory footprint of a neural network while retaining as high an accuracy as possible. Various original methods where shown to be both simple and effective in neural network pruning, with many still being used in practice including magnitude-based pruning and hessian-based pruning. In recent years, a variety of pruning methods have been proposed which have shown progressively greater effectiveness [1]–[9].

Current pruning methods aim to remove the chosen parameters from the neural network suddenly, destroying the information contained within those parameters in a single cut. We hypothesize that this will remove information from the neural network and so make retraining difficult after pruning, particularly at high levels of pruning. To resolve this, we propose *Soft Pruning*: decaying the parameters to be pruned out of the neural network over batches or epochs while training the parameters in the neural network that are not being pruned. This enables the information contained within the pruned parameters to be transferred throughout the neural network before being completely removed, information which would otherwise have been lost had a single cut been used. We hypothesize that this conservation of information will enable lower loss and greater accuracy at higher levels of pruning compared to the conventional method of sudden pruning and then retraining the neural network, which we refer to as *hard pruning*.

First, we show that Soft Pruning results in higher accuracy than hard pruning in the case of classification, particularly at high levels of pruning, achieving above 60% accuracy on a 10-way classification problem where hard pruning is only able to achieve 10% accuracy. We then illustrate the use of Soft Pruning in an image autoencoder, where we apply Soft Pruning to the decoder, showing that Soft Pruning improves on hard pruning in all cases tested, in the case of FashionMNIST, reporting greater than 16% reduction in the loss compared to hard pruning. Finally, we use Soft Pruning for the purpose of latent space dimensionality reduction and show that it is possible to reduce the dimensionality of the latent space of an autoencoder while retaining a low loss. Soft Pruning is able to significantly outperform hard pruning, achieving up to 60.9% reduction in loss after pruning, while also only increasing in loss by 8.4% from the original unpruned neural network when applied using the MNIST dataset.

Latent space operations are becoming increasingly important, with greater emphasis recently placed on the latent space in modern large neural networks, such as the latent diffusion models proposed in recent years [10], [11]. The ability to soft prune the latent space more effectively enables a form of latent space optimisation and fine-tuning, which could be critical in both achieving a well performing neural network as well as reducing the memory footprint of the neural network.

II. Related Work

Initial methods of pruning neural networks were first proposed in the 1990s. This included magnitude-based pruning, removing low magnitude parameters, and using second-order methods to perform pruning. Optimal Brain Damage [12], one of the original works regarding the use of second-order pruning methods, showed that the use of a diagonal hessian approximation enables identifying low-saliency parameters that can be removed from the neural network with minimal effect on the loss. This was improved in another paper published three years later [13], showing the inverse hessian can be used to better identify low saliency parameters, yielding a neural network which produces a lower loss when those parameters are pruned. Due to the simplicity of Hessian-based pruning, both theoretical and practical, this method is still used today.

Magnitude-based pruning is another original method of pruning that has re-gained popularity in recent years [1], [6]–[8], [14], [15], with work showing it is both efficient and effective when applied to large neural networks and difficult problems [16], [17]. In this work, we focus on magnitude-based pruning. This is because our work is not about the method of selecting the parameters to prune, but rather the method of removing them, and magnitude based pruning is both a simple and effective method.

In terms of the method used to remove the selected weights, three main approaches have emerged: One-shot pruning [1]–[4] which involves pruning the neural network once then retraining the neural network from the pruned state, iterative pruning [5]–[8] which involves iteratively pruning and retraining the neural network, and sparsity induced neural networks [5], [9], [17]–[19] which involves specifying a sparsity mask, either at the beginning of training or learning a sparsity mask during training, and training the neural network with this sparsity mask. Another method when seeking a smaller network with a similar architecture is simply to train a smaller neural network on the given task, which has been shown to achieve good results on challenging tasks [20]–[22]. The smaller dense neural networks can be more efficient than the larger pruned network, if structured sparsity is not used; however, this is not a method of pruning and, based on prior work [23], it would be expected to perform worse than using a larger neural network which has been pruned.

Our method of pruning does not fall under any of these categories because each of them imply a single pruning iteration, at the beginning of, iteratively during, or after training, followed by retraining the neural network. Instead, in this work, the neural network is pruned over several iterations by decaying the parameters that are selected to be pruned out of the neural network without retraining the neural network at the end of the process. The closest method to this, and the one against which we compare our work, is one-shot pruning followed by retraining, because every parameter to be pruned is modified at once. Iterative pruning is similar to this method however involves iterative steps of pruning, each step of which prunes different parameters. This is different from taking a snapshot of the weights to prune and then removing those weights in a single cut (one-shot) or, as we propose, by decaying the parameters out of the neural network. Therefore we do not compare our work against iterative pruning, because, for a fair comparason, that would require iterative Soft Pruning which we do not study in this paper.

A similar method to ours has been suggested when pruning for structured sparsity [6], but the authors do not study the difference in the pruning methods, only suggesting that decaying the parameters will likely be better. Here such a study is performed.

III. Method

In this work, we propose slowly decaying parameters out of a neural network, which we term *Soft Pruning*, in contrast to suddenly cutting the parameters out of a neural network as is done in conventional one-shot pruning, which we term *hard pruning*. Several different decay functions can be used for the Soft Pruning operation, and we considered each of those shown in Figure 1. In this paper we use the exponential decay function. This is because in our initial experiments on trivial problems, each of the decay functions was found to reach the same result, with the exponential decay function reaching that result the fastest.

Our method of pruning can can be expressed mathematically as follows. Using W_{ij} as the parameter matrix, M_{ij} as the pruning mask, whose values are 1 to indicate that a parameter is to be pruned and 0 to indicate the parameter should remain, x to represent the decay factor and ΔW_{ij} to represent the gradients of the parameter matrix, a new parameter update rule can be defined by extending gradient descent to allow for reducing the value of chosen (masked) parameters to zero,

$$W_{ij} = W_{ij} \cdot M_{ij} \cdot x + (W_{ij} + \Delta W_{ij}) \cdot (1 - M_{ij}) \quad (1)$$

To choose which parameters to remove, we use the magnitude of the parameter. This is because this method is a popular method in practice and has been shown to be effective and efficient. A more correct but computationally intensive method would be to make use of the Hessian to determine which parameters to remove; however, this cannot be applied generally because such a method can become too computationally intensive to compute for large neural networks. When pruning dimensions of the latent space (section VI) we make use of skeletonization [24] for pruning nodes within the latent space (ie pruning dimensions). For all of the neural networks used in this work, we use the tanh activation function. This is because magnitude-based pruning when using ReLU is equivalent to random pruning when the context of the parameter within the neural network is not considered due to the unbounded nature of the ReLU activation function.

IV. Soft Pruning: Classification

We first perform classification using simple feedforward neural networks, the architectures of which are given in the Appendix. This is trained on three vision based tasks, including the classification of the MNIST handwritten digit dataset [25] (Figure 2), the FashionMNIST clothing dataset [26] (Figure 4) and the CIFAR10 dataset [27] (Figure 5). The hyperparameters and neural network architectures used are given in the Appendix.

As shown in Figure 2, 4 and 5, hard pruning alone is consistently worse than hard pruning with retraining or Soft Pruning, as would be expected. Note that hard pruning and Soft Pruning were both given the same amount of data (1 epoch was used in this experiment) given in the same order, and yet Soft Pruning is able to perform significantly better at higher levels of

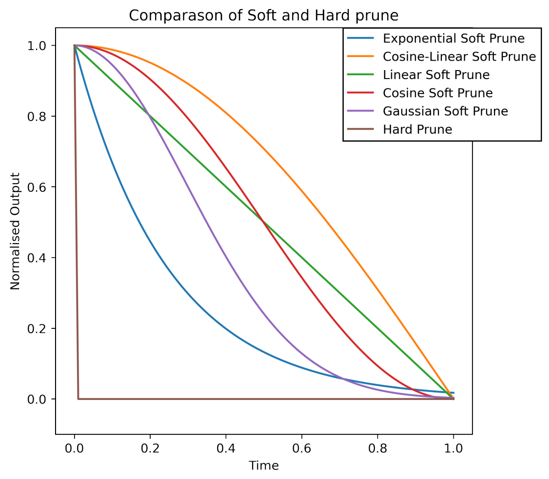

Fig. 1. A sample of Soft Pruning decay functions. In this work, we use the exponential decay function shown in blue. Alternative possible functions are also shown.

pruning. We propose this is because at lower levels of pruning, there are enough nodes such that the information lost from hard pruning can be recovered by retraining. However, at larger amounts of pruning, the performance of hard pruning degrades significantly more than Soft Pruning. This was observed in all tested datasets: MNIST, FashionMNIST and CIFAR10. Furthermore, a plateau is observed when applying Soft Pruning to the MNIST and CIFAR10 experiments. We suggest that this is caused by retaining the features that give as high accuracy as possible, while disregarding other features. These features could be simple features, but they may be expressed by complicated parametric configurations within the trained neural network. These parametric configurations are forcibly compressed by the application of Soft Pruning, and it is only when they cannot be expressed in the few nodes remaining that there is a sudden degradation in the accuracy. Notice that in the case of CIFAR10, the accuracy is particularly low, staring at 50% accuracy when pruning 70% of the neural network and expressing a Soft Pruning plateau at approximately 35% accuracy, but this is higher than random guessing. In this case, the neural network configuration may be too small or too simple to represent the CIFAR10 dataset with high accuracy. However a platau is observed when using Soft Pruning where no such platau is observed when using hard pruning, which suggests Soft Pruning is able to preserve information which hard pruning cannot, supporting our hypothesis.

In the case of FashionMNIST, a continuous degradation is observed when using Soft Pruning as the level of pruning is increased, and there is no platau. We believe this is because the FashionMNIST dataset does not contain classes which appear similar to each other, many of the classes could appear distinct to a neural network classifier. And so, because each class is as complicated as every other class, there is a gradual decrease in performance as the classifier keeps as much information as possible to perform the classification to as high an accuracy as possible.

In the case of the MNIST dataset, higher than 60% accuracy is retained until 97% of the parameters are pruned when using Soft Pruning. To further analyze this, we plot a confusion matrix for the classifications of the 95% pruned model, shown in Figure 3. The Soft Pruning is shown to retain enough information to correctly predict 1 and 7 well, but loses accuracy on classes 3 and 8. This is unexpected because intuitively it would be simpler to merge similar classes and retain distinctions between different classes; however, in this case the opposite appears to be happening. The classes with more similar, but simpler, features are retained the best, with those classes having the highest accuracy after pruning, and the classes with the most distinctive features, such as 3 and 8, are merged into the classes with simpler features.

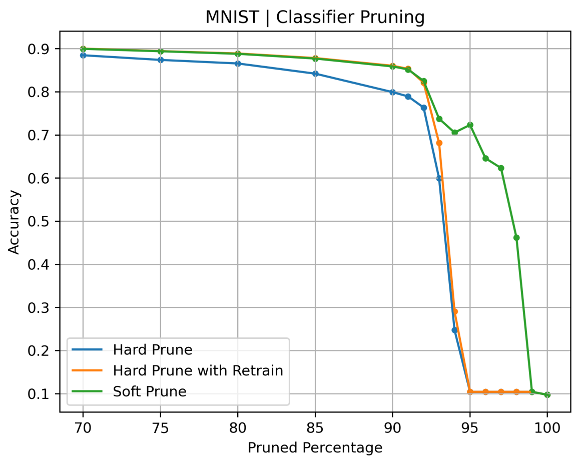

Fig. 2. The results of Soft and hard pruning on the MNIST dataset. The network architecture and hyperparameters used for the training are given in the Appendix. Notice there is a platau present in Soft Pruning which is missing in hard pruning with retraining; in this task, Soft Pruning can achieve better performance than hard pruning with retraining at high levels of pruning. The initial number of parameters is 211,626. Pruned percentage is the percentage of parameters that are pruned from the neural network.

V. Soft Pruning: Auto-encoder

We next perform a comparison of Soft Pruning and hard pruning on the case of an autoencoder encoding visual information. We make use of the same datasets as in the previous section: MNIST, FashionMNIST and CIFAR. In this experiment, one epoch of training data was used for both Soft Pruning and hard pruning with retraining. Hard pruning alone was also tested on these experiments but gave a significantly higher loss value than hard pruning with retraining or Soft Pruning, at all levels of pruning. Therefore the results of hard pruning alone are excluded from the figures of this experiment. In all cases tested, Soft Pruning is able to outperform hard pruning on loss. At levels of pruning between 80%-90%, a 0%-13.1% improvement in loss compared to hard pruning is observed; at higher levels of pruning, this increases up to a 17.6% improvement in loss. Given that the same parameters are ultimately pruned, this suggests that Soft Pruning is able

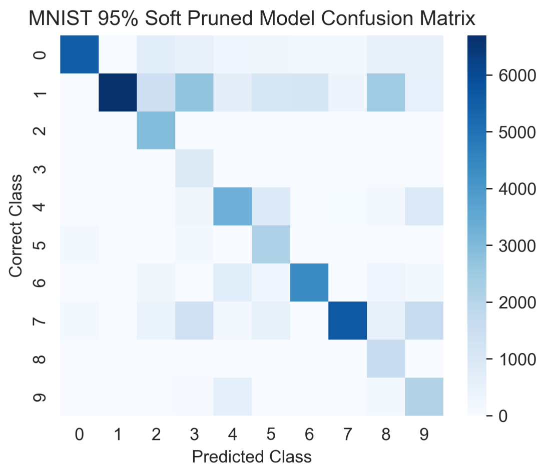

Fig. 3. A confusion matrix of the 95% soft pruned neural network applied to the MNIST training dataset.

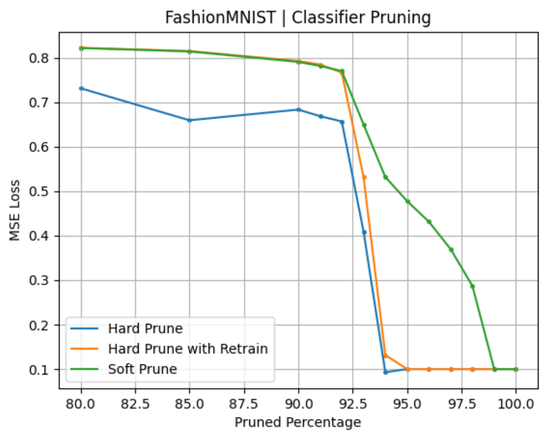

Fig. 4. The result of pruning a neural network on the FashionMNIST dataset. The neural network used is given in the Appendix. The initial number of parameters is 211,626.

TABLE I
SOFT PRUNING RESULTS | LINEAR CLASSIFICATION

Dataset	Accuracy (%)		
Pruned Percentage	**90%**	**95%**	**97%**
MNIST			
Hard Prune	79.9	10.4	10.4
Hard Prune & Retrain	**86.1**	10.4	10.4
Soft Prune	85.9	**72.3**	**62.3**
FashionMNIST			
Hard Prune	68.4	10.0	10.0
Hard Prune & Retrain	**79.3**	10.0	10.0
Soft Prune	79.1	**47.8**	**37.0**
CIFAR-10			
Hard Prune	30.3	10.0	10.0
Hard Prune & Retrain	**48.1**	10.0	10.0
Soft Prune	47.7	**37.4**	**35.4**

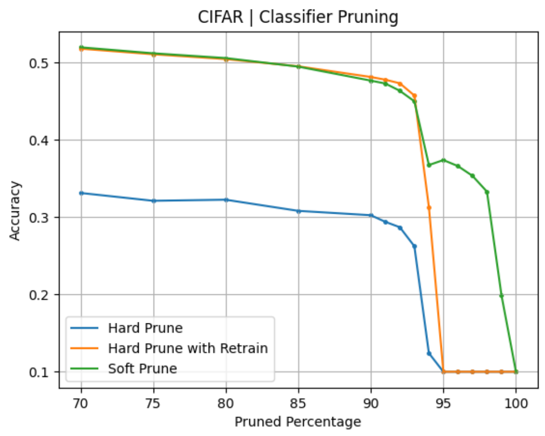

Fig. 5. The result of pruning a classifier on the CIFAR10 dataset. Notice a platau is observed at high levels of pruning, similar to the FashionMNIST dataset. The initial number of dense parameters is 84,522. Here a convolutional feature extractor is used on the input of the neural network.

to retain information contained within the pruned parameters, supporting our hypothesis. In the case of the MNIST autoencoder (Figure 6, left), Soft Pruning is marginally better, achieving 0%-1.4% lower loss up to 90% parameters pruned, then achieving up to 9.2% lower loss, where the loss begins to increase rapidly. Comparing Soft Pruning to hard pruning on the FashionMNIST dataset (Figure 6, center) shows clearly that Soft Pruning can achieve significantly lower levels of loss as the level of pruning increases. This dataset is visually harder to reproduce using an autoencoder because there are more features in each image, hence, using Soft Pruning allows for up to 16.7% lower loss by preserving the information present in the original unpruned neural network, with consistently greater than 10% difference in the loss above 90% parameters pruned. In the case of CIFAR(Figure 6, right), up to 90% of the parameters pruned there is again marginal difference between Soft Pruning and hard pruning, where Soft Pruning is lower by 0%-1.3%. At higher levels of pruning however this difference increases up to 17% lower loss.

Each of these experiments shows that Soft Pruning is able to match or outperform hard pruning. Even though hard and Soft Pruning were given the same amount of retraining data, Soft Pruning is able to preserve information from the unpruned neural network and so is able to consistently report lower loss than hard pruning at high levels of pruning.

In each experiment, hard pruning is able to match Soft Pruning until a point. We suggest this is because, until this point, any information that is lost can be recovered by retraining the neural network, making use of the remaining parameters to achieve a result equivalent to Soft Pruning. When there are insufficient parameters remaining, the performance of hard pruning degrades significantly more than Soft Pruning.

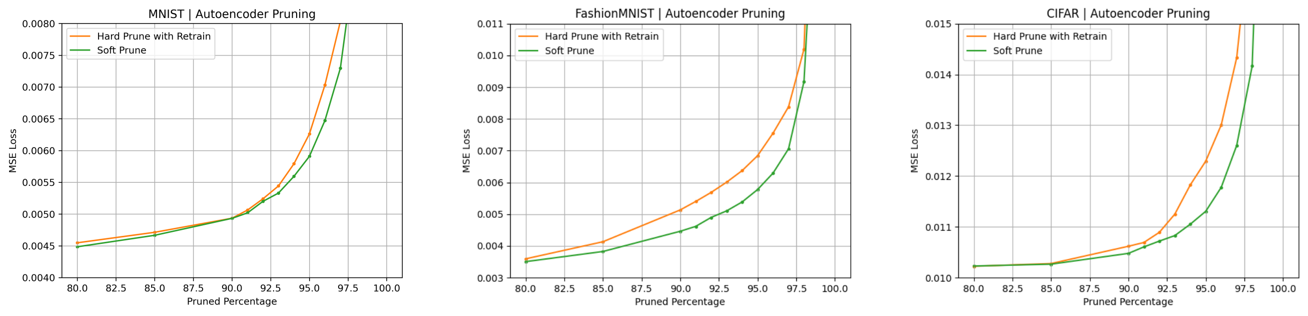

Fig. 6. The results of pruning a visual autoencoder using either Soft Pruning or hard pruning on the MNIST (left), FashionMNIST (center) and CIFAR (right) datsets. Soft Pruning is able to either match or outperform hard pruning in all experiments. Only the decoder is pruned. The initial number of parameters in the decoder of the autoencoder is 590,912.

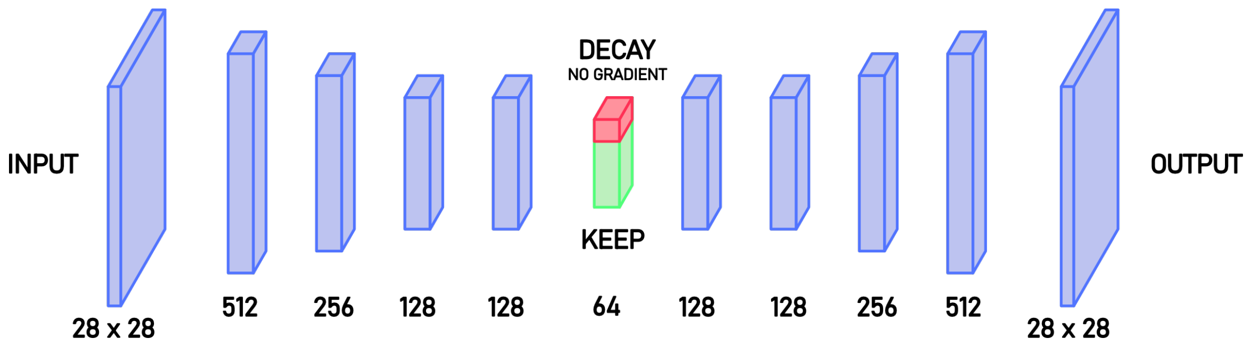

Fig. 7. An illustration of the neural network used for the latent space dimensionality reduction applied to the MNIST and FashionMNIST datasets. Blue layers indicate layers that were not changed, red indicates nodes which are being decayed out of the neural network, and green indicates nodes which can update their parameters (using the gradient provided by backpropagation.)

VI. Soft Pruning: Latent Space Dimensionality Reduction

Finally, we study the effect of Soft Pruning applied to latent space dimensionality reduction. It can be beneficial to keep a latent space small because this can reduce complexity of interacting with it and can increase processing speed.

In order to determine which nodes in the latent space to prune, we make use of the skeletonization algorithm [24]. This calculates the saliency of each node in a neural network by taking the derivative of the loss with respect to a multiplicative parameter on each node. Putting this mathematically, the saliency of a node can be calculated by first replacing every node n_i with a new value

$$n_i = a_i \cdot m_i \tag{2}$$

The saliency of a node s_i can then be found by taking the gradient of the loss E with respect to the parameter a_i,

$$s_i = \frac{\partial E}{\partial a_i} \tag{3}$$

The magnitude of this measure indicates the saliency, s_i, of a node, n_i, within the neural network, where lower values indicate the node has a smaller effect on the loss. When removing the nodes, we sort the nodes by saliency and select the top-n lowest saliency nodes to remove. To remove the node, we prune the parameters on either side of the node, which has the effect of pruning the node.

We test this method on the same datasets as used in the previous sections: MNIST, FashionMNIST and CIFAR. In all cases, Soft Pruning is shown to outperform hard pruning, often by a significant margin. In the case of MNIST and Fashion-MNIST, we reuse the same autoencoder from the previous section, but freeze every parameter except the parameters on the input and output of the latent space (illustrated in Figure 7). We then identify the nodes to prune within the latent space using skeletonisation and prune the parameters on either side of the selected nodes. hard pruning alone produced results that are particularly poor and appear off the graph and so are not including in the figures for this experiment. In this experiment, five epochs of training data are used for both Soft Pruning and hard pruning with retraining.

As shown in Figure 8 (MNIST) and Figure 10 (Fashion-MNIST), Soft Pruning is able to retain a significantly lower loss than hard pruning with retraining as the pruning level increases. Furthermore, the use of Soft Pruning caused the loss to only increase marginally up to 95% of the nodes pruned from a 64 node latent space. Given the same nodes were pruned for both hard pruning and Soft Pruning, this clearly shows Soft Pruning is able to conserve the information within the nodes and enable achieving a consistently lower loss

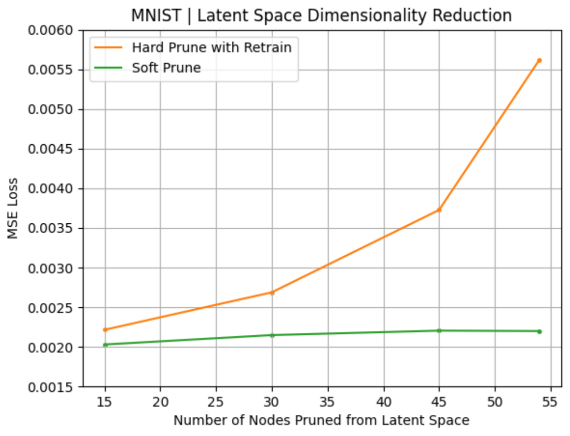

Fig. 8. The results of progressively decreasing the dimensionality of the latent space of the autoencoder for the MNIST dataset. The latent space has 64 nodes before pruning.

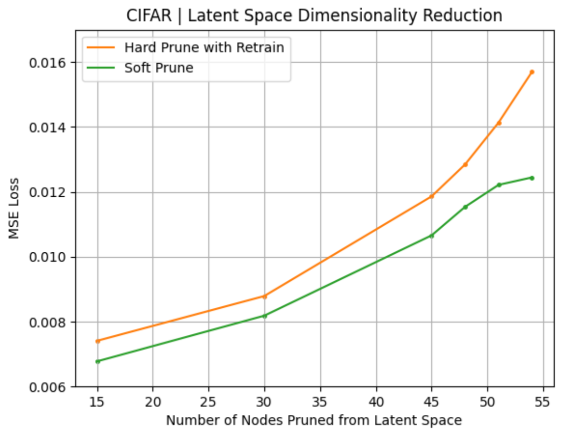

Fig. 9. The results of progressively decreasing the dimensionality of the latent space of the autoencoder for the CIFAR dataset using either soft or hard pruning. The latent space has 64 nodes before pruning.

than hard pruning. In the case of MNIST, the loss from Soft Pruning is found to be up to 60.9% lower than the loss from hard pruning with retraining. In the case of FashionMNIST, the loss when using Soft Pruning is up to 35.8% lower than when using hard pruning with retraining. Furthermore, the loss at each pruning level remains close to the loss of the unpruned model when using Soft Pruning, with MNIST only increasing by 8.4% at 95% pruned, in comparison to the 154% increase found using hard pruning with retraining. A similar result is also found on the FashionMNIST dataset, with the loss increasing by only 8.3% at 95% pruned, compared to a 90% increase using hard pruning with retraining. This result supports our hypothesis that the use of Soft Pruning enables conserving useful data within the pruned parameters, enabling a significantly lower loss, compared to hard pruning, as the level of pruning increases.

In the case of CIFAR, we use a convolutional encoder and decoder with a dense layer inbetween containing a 64 dimension latent space. The dense layer is used because this paper does not study pruning the weights of a convolutional layer, which, due to the nature of a convolution operation, could have different ramifications compared to pruning a dense layer. As shown in Figure 9, Soft Pruning is able to consistently achieve over 10% lower loss than hard pruning with retraining, reaching a difference of 20.7% at 95% of the parameters pruned.

VII. Conclusion

In this paper we propose Soft Pruning and apply it to three different visual tasks: classification, autoencoding and latent space dimensionality reduction. In all instances, Soft Pruning is shown to match or outperform hard pruning, with significant improvements in performance appearing at high levels of pruning. Given this difference in performance despite having the same amount of retraining data as hard pruning, our work suggests that Soft Pruning is able to better preserve the parametric information present in the original unpruned neural network, achieving a lower loss after pruning, supporting our hypothesis.

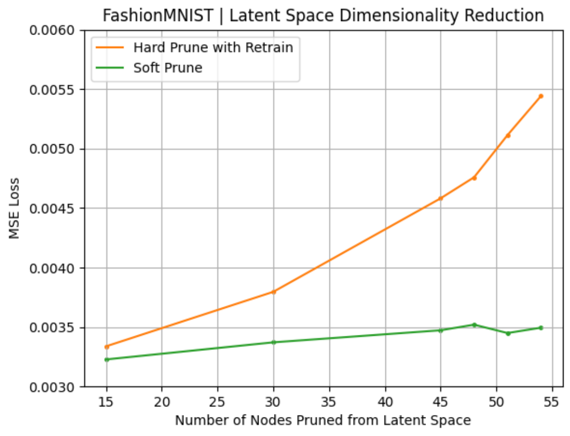

Fig. 10. The results of progressively decreasing the dimensionality of the latent space of the autoencoder for the FashionMNIST dataset using either soft or hard pruning. The latent space has 64 nodes before pruning.

VIII. Future Work

In this work we apply our method to feedforward dense layers in a neural network, the generalisation of the results presented in this paper to other types of neural network layers and architectures has yet to be explored. Pruning a convolutional neural network using larger datasets would be expected to show similar results to those presented in this work but such a study could yield additional insight into Soft Pruning applied in this application. Similarly recurrent neural networks are not studied in this work and a study of Soft Pruning applied to a recurrent task, such as language modeling or reinforcement learning, could show further improvements when using this method.

TABLE II

Soft Pruning Results
Latent Space Dimensionality Reduction

Dataset	Loss ($\times 10^{-3}$)		
Pruned Percentage	**25%**	**50%**	**75%**
MNIST			
Hard Prune	5.70	11.52	17.56
Hard Prune & Retrain	2.21	2.69	3.73
Soft Prune	**2.02**	**2.15**	**2.20**
FashionMNIST			
Hard Prune	8.15	15.9	22.9
Hard Prune & Retrain	3.34	3.80	4.58
Soft Prune	**3.23**	**3.37**	**3.47**
CIFAR			
Hard Prune	13.08	19.06	25.91
Hard Prune & Retrain	7.41	8.79	11.86
Soft Prune	**6.77**	**8.18**	**10.66**

Due to our method performing continuous parametric optimisation, it should also be possible to optimise wanted features into the latent space during pruning or optimise out unwanted features during pruning, leaving a fine-tuned latent space for a given task. This could be beneficial where only a fraction of the space from the original unpruned neural network is needed.

IX. Reproducibility Statement

All the code in this paper is hosted online at *[Link hidden for peer review]*. Random seeds are used to ensure the results are reproducible.

Appendix
Network Architectures & Hyperparameters

For every neural network presented, the tanh activation function is applied after every intermediary dense or convolutional layer excluding the flatten, reshape, sigmoid layers. Additionally no activation function is applied before the softmax layers.

A. Classifier

For training, the stochastic gradient descent (SGD) optimiser with a learning rate of 0.01 and no momentum or weight decay was used. For MNIST and FashionMNIST, both datasets have 60,000 images in total, in this work 50,000 images were used for training the neural network. The 10,000 were used to ensure the results generalise to the test dataset. The neural network architectures used in the classification task are given in Table III. Before input to the neural network, the images of every dataset were channel-wise normalised and no other data augmentation was used.

B. Autoencoder

The neural network architectures used are given in Table V. For training, the AdamW optimiser is used with a learning rate of 0.001 and weight decay of 0.0001. For retraining, both in the Soft Pruning and hard pruning case AdamW is used again with the same hyperparameters. The same datasets used in the classifier tasks were used here, with the same training/test splits. The same data augmentation was used as in the Classifer section.

C. Latent Space Dimensionality Reduction

The neural network architectures used for the latent space dimensionality reduction experiments are given in Table IV. The same optimiser and data augmentation is used as in the Autoencoder section.

TABLE III

Classification

MNIST & FashionMNIST	**CIFAR**
FLATTEN	FLATTEN
Dense(256)	Conv2D(8, 3x3)
Dense(128)	Max Pool(2x2)
Dense(64)	Conv2D(16, 3x3)
Dense(32)	Max Pool(2x2)
Dense(10)	Dense(576)
SOFTMAX	Dense(128)
	Dense(64)
	Dense(32)
	Dense(10)
	SOFTMAX

TABLE IV

Autoencoder (Latent Space Dimensionality Reduction)

MNIST & FashionMNIST	**CIFAR**
FLATTEN	FLATTEN
Dense(512)	Conv2D(64, 3x3)
Dense(256)	Conv2D(128, 3x3)
Dense(128)	Max Pool(2x2)
Dense(128)	Conv2D(128, 3x3)
== Dense(64) ==	Conv2D(128, 3x3)
Dense(128)	Max Pool(2x2)
Dense(128)	Conv2D(128, 3x3)
Dense(256)	FLATTEN
Dense(512)	== Dense(64) ==
RESHAPE(28x28)	RESHAPE(8, 3x3)
SIGMOID	ConvTranspose2D(128, 3x3)
	SCALE(2)
	ConvTranspose2D(128, 3x3)
	ConvTranspose2D(128, 3x3)
	SCALE(2)
	ConvTranspose2D(128, 3x3)
	ConvTranspose2D(3, 3x3)
	SIGMOID

TABLE V
AUTOENCODER

MNIST & FashionMNIST & CIFAR
FLATTEN
Dense(512)
Dense(256)
Dense(128)
Dense(128)
== Dense(64) ==
Dense(128)
Dense(128)
Dense(256)
Dense(512)
RESHAPE(28x28)
SIGMOID

REFERENCES

[1] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rJl-b3RcF7

[2] N. Lee, T. Ajanthan, and P. Torr, "SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=B1VZqjAcYX

[3] A. Mishra, J. A. Latorre, J. Pool, D. Stosic, D. Stosic, G. Venkatesh, C. Yu, and P. Micikevicius, "Accelerating sparse deep neural networks," 2021.

[4] J. Pool and C. Yu, "Channel permutations for n:m sparsity," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 13 316–13 327. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/6e8404c3b93a9527c8db241a1846599a-Paper.pdf

[5] U. Evci, F. Pedregosa, A. Gomez, and E. Elsen, "The difficulty of training sparse neural networks," in *ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena*, 2019. [Online]. Available: https://openreview.net/forum?id=SyeyPEH23N

[6] S.-C. Kao, A. Yazdanbakhsh, S. Subramanian, S. Agrawal, U. Evci, and T. Krishna, "Training recipe for n:m structured sparsity with decaying pruning mask," 2023. [Online]. Available: https://openreview.net/forum?id=bMXueK316u

[7] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/2823f4797102ce1a1aec05359cc16dd9-Paper.pdf

[8] Z. Yao, S. Cao, W. Xiao, C. Zhang, and L. Nie, "Balanced sparsity for efficient dnn inference on gpu," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 5676–5683, Jul. 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4512

[9] J. Frankle, G. K. Dziugaite, D. Roy, and M. Carbin, "Pruning neural networks at initialization: Why are we missing the mark?" in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=Ig-VyQc-MLK

[10] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 10 684–10 695.

[11] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach, "SDXL: Improving latent diffusion models for high-resolution image synthesis," in *Submitted to The Twelfth International Conference on Learning Representations*, 2023, under review. [Online]. Available: https://openreview.net/forum?id=di52zR8xgf

[12] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed., vol. 2. Morgan-Kaufmann, 1989. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf

[13] B. Hassibi and D. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems*, S. Hanson, J. Cowan, and C. Giles, Eds., vol. 5. Morgan-Kaufmann, 1992. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1992/file/303ed4c69846ab36c2904d3ba8573050-Paper.pdf

[14] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrelln, "Rethinking the value of network pruning," in *NIPS 2018 workshop on Compact Deep Neural Networks with industrial applications*, 2019. [Online]. Available: https://openreview.net/pdf?id=r1eLk2mKiX

[15] A. Renda, J. Frankle, and M. Carbin, "Comparing rewinding and fine-tuning in neural network pruning," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=S1gSj0NKvB

[16] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf

[17] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," *arXiv e-prints*, vol. arXiv:1902.09574, 2019. [Online]. Available: https://arxiv.org/abs/1902.09574

[18] M. Wortsman, A. Farhadi, and M. Rastegari, "Discovering neural wirings," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/d010396ca8abf6ead8cacc2c2f2f26c7-Paper.pdf

[19] T. Dettmers and L. Zettlemoyer, "Sparse networks from scratch: Faster training without losing performance," 2019.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[21] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019. [Online]. Available: https://openai.com/research/better-language-models

[22] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer, "Opt: Open pre-trained transformer language models," 2022.

[23] S. G. Michael H. Zhu, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," 2018. [Online]. Available: https://openreview.net/forum?id=S1lN69AT-

[24] M. C. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed., vol. 1. Morgan-Kaufmann, 1988. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1988/file/07e1cd7dca89a1678042477183b7ac3f-Paper.pdf

[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[27] (2009) Learning multiple layers of features from tiny images. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf