Pruning Deep Equilibrium Models

Hoang Pham[*]
FPT AI Lab, Vietnam
HoangPV19@fsoft.com.vn

Tuc Nguyen[*]
FPT AI Lab, Vietnam
TucNV2@fsoft.com.vn

The-Anh Ta
School of Engineering
Tan Tao University, Long An, Vietnam
anh.ta@ttu.edu.vn

Dung D. Le
College of Engineering and Computer Science
VinUniversity, Ha Noi, Vietnam
dung.ld@vinuni.edu.vn

Long Tran-Thanh
Department of Computer Science
University of Warwick, UK
long.tran-thanh@warwick.ac.uk

Abstract

We study the effects of pruning on deep equilibrium models - a new kind of neural network layer. Our experimental results show that pruning methods can be applied successfully to these new layers and help to reduce their training and inference complexity. Our investigation also raises several questions about pruning deep equilibrium models, for example, i) what are the correct criteria for pruning this new layer compared to the usual scores widely used for explicit neural networks? ii) how to design sparse parametrizations for deep equilibrium models for better effectiveness and efficiency?

1 Introduction

Implicit layers is a new type of neural network layer recently introduced in [1, 7, 2, 9]. Implicit layers have been increasingly gaining attention from the research community thanks to many of their advantages compared to traditional neural network layers [2]. There have been many effort in the literature to understand the working mechanism of implicit layers such as [27, 9, 20, 28] on well-posedness, [14, 11] on optimization process, and [17, 13, 8, 26] on robustness and certification, etc. The potential of implicit layers has been demonstrated by competitive performance with usual neural networks and other advantages on many settings, from computer vision [3, 5], time series [21], generative modeling [12] to optimization [18], logical reasoning [25], games theory [16], etc.

In this paper, we initiate the investigation of pruning methods for implicit layers. We restrict ourself here to one main class of deep equilibrium models (DEQ) [2, 9], though similar studies can be carried out for other implicit model classes like neural ordinary differential equations [7], or continuous normalizing flows [22]. A general deep equilibrium model consists of 2 parts: the DEQ layer and the classifier. On an input $x \in \mathbb{R}^d$, a DEQ layer outputs $z^\star \in \mathbb{R}^n$ defined as follows. Given an elementwise nonlinearity $\sigma : \mathbb{R}^n \to \mathbb{R}^n$, hidden unit weights $W \in \mathbb{R}^{n\times n}$, input-injection weights $U \in \mathbb{R}^{n\times d}$ and bias term $b \in \mathbb{R}^n$, the output vector is defined by

$$z^\star = \sigma(Wz^\star + Ux + b), \tag{1}$$

which is approximated in the forward pass by fix-point iteration methods [2, 27]. The classifier is a linear layer or a multilayer perceptron applied to the feature $z^\star$. We only consider linear classifier that

[*]Joint first authors, equally contributed.

[2]http://implicit-layers-tutorial.org/introduction/

on DEQ layer's output $z^\star \in \mathbb{R}^n$ gives $\hat{y} = Cz^\star + Dx + c$, where $C \in \mathbb{R}^{d_{out} \times n}$, $D \in \mathbb{R}^{d_{out} \times d}$, $c \in \mathbb{R}^{d_{out}}$. In the backward pass, gradient of loss function is computed by implicit differentiation [2].

A priori, the well-posedness of a DEQ layer is not guaranteed, that is the vector $z^\star$ defined in Equation (1) might not exist and be unique for each input vector x. Additional conditions on weight matrices of DEQs have been proposed in [27, 9] to ensure the well-posedness. Since we are interested in network pruning, it might also happen that while the full DEQ is well-posed, the subnetwork obtained after pruning will be ill-posed [13]. Because of such problems, we will focus on a main subclass of DEQs called Monotone DEQ (monDEQ) introduced in [27] for which the well-posedness is always guaranteed for pruning. We also perform experiments on some non-monotone DEQs whose training stability probably induces well-posedness for pruning, but no theoretical guarantee is possible.

Monotone DEQ (monDEQ) [27] is different from the general DEQ in the parametrization of the hidden unit weights $W \in \mathbb{R}^{n \times n}$:

$$W = (1 - m)I - A^T A + B - B^T, \tag{2}$$

where $m > 0$ is a hyperparameter, $A, B \in \mathbb{R}^{n \times n}$ are trainable weights. Apart from the parametrization of W in terms of A, B, monDEQ is the same as general DEQ. The parametrization in Equation (2) ensures that $z^\star$ exists and is unique by general results from convex optimization. Splitting operator equilibrium solving methods like forward-backward or Peaceman-Rachford are used as blackboxes for finding $z^\star$ [27]. Note that for monDEQs, pruning weights in A, B means replacing A, B by $m_A \odot A, m_B \odot B$ with 0-1 masking matrices m_A, m_B, which still gives another monDEQ parametrization for the subnetwork obtained from applying any pruning method. Thus, the existence and uniqueness of output is always ensured.

Other problems with DEQs concerns their training gradient stability and long inference time. Some regularization (Jacobian regularization [4], sparse fixed-point correction [5]) and initialization ([6]) strategies have been proposed to deal with such problems which are crucial for possible deployment. Our investigation on pruning DEQs suggests that pruning techniques can be another solution for stability and inference time of DEQs. In addition to comparisons of sparse monDEQ and dense monDEQ with the same number of parameters, we observe that at high sparsity, one weight matrix in the monotone parametrization is usually pruned almost completely, while the performance of the sparse DEQ is mostly the same with the dense model. This leads us to propose a new parametrization for monDEQ by simply removing a weight matrix from its definition. Our sparse parametrizations are shown by experiments to achieve similar performance with the original monDEQ parametrization, while reduce running time significantly.

Our contributions in this short paper are two-fold. *First,* we perform extensive experiments on pruning DEQs by pruning at initialization methods; make comparison with dense DEQs of the same size to show the specialities of sparse DEQs; we also demonstrate the benefits of pruning DEQs in reducing running time. *Second,* we propose a simple reduced parametrization of monDEQ based on observing sparsity patterns of pruning, and show that the new parametrization still achieve the same performance as the original monDEQ while reduce significantly the computational complexity.

2 Pruning Deep Equilibrium Models

In this section, we investigate the effection of the five pruning at initialization methods, which are widely used in explicit models, on DEQ and monDEQ models with different architectures and datasets. For further detail of the pruning methods are presented in the Appendix B. We first describe the experimental setup, then we show the results of sparse monDEQ models and compare them to corresponding dense monDEQ models. We also discuss the convergence rate of the fixed-point solver in sparse monDEQ models.

2.1 Results of pruning monDEQ

Experimental setting To test the efficiency of sparse monDEQ models, we evaluate the performance of two architectures (fully connected, single convolution) on three datasets (MNIST, SHVN, CIFAR-10) which are used in [27]. Notice that the monDEQ models comprise the implicit layer and the explicit classifier. Here, we only prune the implicit layer with the assumption that the sparse implicit layer still can achieve adequate representations. Five pruning at initialization methods: Random, Magnitude, SNIP [15], GraSP [24] and SynFlow [23] are used with the sparsity ratio in

range $s = \{10, 50, 70, 90, 95, 97, 99, 99.9\}$. To make fair comparisons, we set the hyperparameters similar to the ones in [27] and run all experiments in three different seeds.

Results of pruned monDEQ In this part, we investigate the performance of pruned monDEQ models with different pruning methods. Overall, pruned models can achieve comparable results to dense models with sparsity smaller than 70% (Figure 1a). When the sparsity gets higher, the performance of single convolution monDEQ models slightly degrades in all pruning methods except random. This indicates that implicit layers also have subnetworks that produce competitive performance with the dense ones. In addition, in the case of extreme sparsity ($s = 99.9\%$) the accuracy of the pruned models decrease significantly. The reason is that the implicit layer does not have enough parameters to learn the representation.

In general, random pruning shows weak efficiency compared with others (Figure 1a). In the case of the monDEQ with single convolution architecture, randomly pruning reduces accuracy enormously when the sparsity is larger than 97% while other pruning schemes still remain stable. Besides, Magnitude pruning consistently performs better than other methods, which is unusual in pruning with explicit networks (except the case using fully connected layers in which the weights are initialized by Uniform distribution with the range determined by fan-in. With sparisty higher than 70%, Magnitude pruning prunes all the parameters in the first layer leading to the accuracy dropping). This behavior seems to indicate that gradient-based methods are not suitable for pruning with implicit layers. This raises the question of *what is the criteria to keep connections in implicit layers?*.

Compare forward iteration between pruned monDEQ and dense monDEQ. Apart from performance, the convergence rate of the fixed points solver is one of the most important problems for DEQ models. Figure 1b illustrates the number of forwarding iterations in finding fixed points with different pruning methods and sparsity ratios on different settings. Figure 1b shows that the convergence rates depend strongly on the sparsity ratio of pruned networks. We see that when the sparsity increases, the smaller number of forwarding interactions is required. It seems that pruning methods act as a regularizer which reduces the norm of implicit layers. As suggested in [4], when matrix norms of weights in implicit layers decrease, the solver becomes more stable leading to faster convergence.

Comparison between sparse and dense monDEQ. In this experiment, we examine the performance of sparse monDEQ models compared with the dense models which have the same number of parameters. Since we only prune the implicit layers, the classifier is kept fixed which has a large number of parameters. To make a fair comparison, we calculate the effective parameters in sparse monDEQ models, then design the dense models such that they have the same number of parameters. We compare these two kinds of models with single convolution monDEQ on the CIFAR-10 dataset, the sparse monDEQ models are produced by SNIP and Magnitude pruning methods.

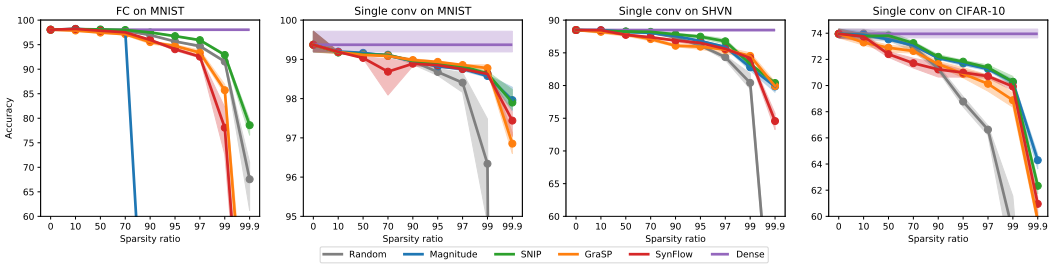
Figure 2a and 2b show that with a lower sparsity ratio, the sparse monDEQ achieves higher accuracy than the corresponding dense monDEQ. When the sparsity is larger than 99%, the performance of sparse monDEQ decreases dramatically since the implicit layers do not have adequate parameters to solve for good fixed points. Meanwhile, the accuracy of dense monDEQ remains stable because of the larger parameters in implicit layers which can be better in generalization.

In addition, with the same sparsity, SNIP has more effective parameters in the classifier leading to larger dense counterparts (see Appendix D.1). That is why though sparse models produced by SNIP and Magnitude have similar performance, the dense models in Figure 2a have better performance than the others in Figure 2b.

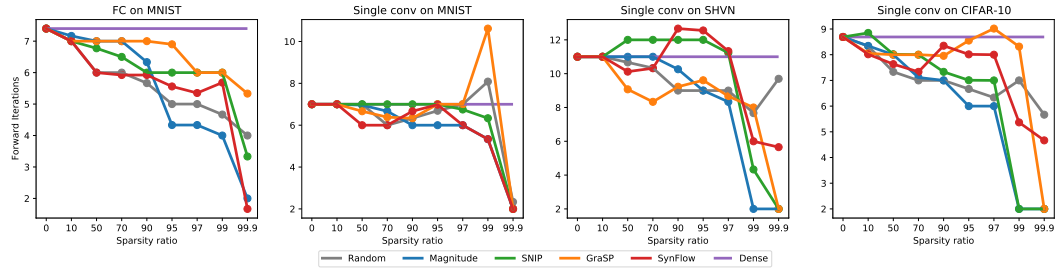
2.2 Results of pruning DEQ

To examine the power of the different pruning methods in DEQ, we move toward applying pruning to resnetDEQ [3] which is an example of non-monotone DEQ. We only prune the implicit layers in the resnet DEQ. Similar to doing pruning on normal deep neural networks, sparsification and accuracy of pruned DEQ model follow Occam's hill [19]. Specifically in Figure 2c, with the small sparsity (less than 10%), the accuracy increases due to the reduction of learned noise. Then, the model reaches an often extended range of sparsities (from 10% to 70%) where the performance remains stable with slight decreases. Eventually, with high sparsity (more than 70%), the generalization performance

[3] http://implicit-layers-tutorial.org/deep_equilibrium_models/



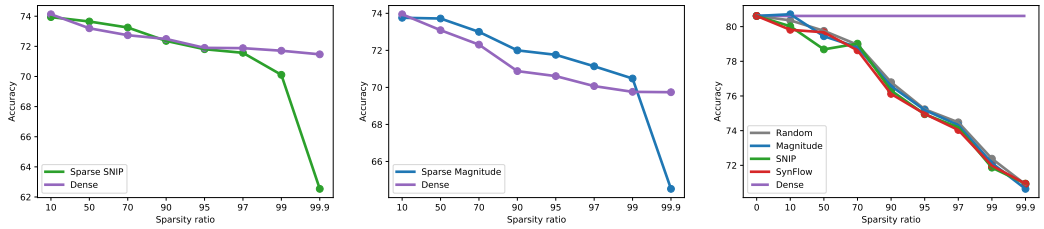
(a) The Sparsity Accuracy of different pruning at initialization methods on monDEQ



(b) The Sparsity-Forward Iterations of different pruning at initialization methods on monDEQ

Figure 1: Sparsity-Accuracy and Sparsity-Forward Iterations of different pruning methods on monDEQ.

quickly degrades. These phenomenon one more time suggest that there are exist subnetworks in DEQ models which can achieve similar performance to the origin one.



(a) Dense vs Sparse monDEQ by SNIP (b) Dense vs Sparse monDEQ by Magnitude (c) Sparsity-Accuracy with resnetDEQ

Figure 2: The first two figures (a and b) illustrate the accuracy of sparse monDEQ models compared to dense monDEQ models with the same number of effective parameters on the CIFAR-10 dataset pruned by SNIP and Magnitude, respectively. The last figure (c) shows the sparsity-accuracy of pruning at initialization methods with the DEQ model on the CIFAR-10.

3 An application: reduced parameterizations for monDEQ

In this section, we investigate the role of the three matrices U, A, B in the implicit layer of the monDEQ models (2). We first show the sparsity of each matrice after pruning. Then, we do ablation studies when sequentially removing components.

3.1 Sparsity patterns in weight matrices of monDEQ after pruning

When pruning the model, we observe that with higher sparsity, the matrix A and B become sparser. Table 1 is the sparsity in each component of monDEQ models after pruning 90% parameters by SNIP. Surprisingly, matrix A is almost pruned in all architectures on all datasets. Meanwhile, almost the weights in matrix U are kept in single convolution models, particularly, around 94%. Figure D1 visualizes heatmap of the three matrices U, A, B in the Equation 2. This raises the question that *how removing A and B affects the performance of the monDEQ models.*

3.2 Reduced parametrization for monDEQ

To answer the above question, we do ablation studies on either removing A or removing B, or removing both. Table 2 shows that when removing A, the performance slightly improves in all

Table 1: Sparsity (%) in each components of monDEQ models after pruning 90% by using SNIP

	MNIST		SHVH	CIFAR-10
	FC	Single conv	Single conv	Single conv
All	90.00	90.00	90.00	90.00
U	90.15	6.10	6.30	5.81
A	94.50	99.92	99.98	99.99
B	84.11	81.63	83.12	83.12

Table 2: Accuracy (%) of monDEQ models when removing components

	MNIST		SHVH	CIFAR-10
	FC	Single conv	Single conv	Single conv
Full	98.09	99.19	88.56	73.64
Remove A	98.20	99.22	88.73	74.12
Remove B	98.06	98.86	86.65	70.80
Remove A & B	97.97	98.50	83.65	70.35

settings. While only removing B, the accuracy reduces significantly in the CIFAR-10 dataset (from 73.64% to 70.80%), slightly drops in SHVN and MNIST (around $1-2\%$). Discarding both A and B leads to matrix W is a diagonal matrix that results in poor representation of fixed points. Consequently, the accuracy drops most severe in all models and datasets.

The results in this section indicate that the matrix A in Equation(2) is not always necessary for the monotone property and the performance of monDEQ. This poses an interesting question: *how to design sparse parametrizations for deep equilibrium models for better effectiveness and efficiency?*

4 Conclusions

This paper reports an initial investigation of pruning methods for deep equilibrium models. We demonstrate empirically that pruning can be done efficiently, and help to reduce the training and inference cost of this DEQ layer. Observations on sparsity patterns of weight matrices after pruning lead to a simplification in the monotone DEQ parametrization. Pruning also reveals various differences in the DEQ layer compared to classic explicit neural network layers and raises problems that are helpful in understanding the working mechanism of this new kind of neural network layers. The upcoming full version of this paper will further investigate various aspects of DEQs pruning, towards a more thorough understanding of the problem.

References

[1] B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

[2] S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[3] S. Bai, V. Koltun, and J. Z. Kolter. Multiscale deep equilibrium models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[4] S. Bai, V. Koltun, and J. Z. Kolter. Stabilizing equilibrium models by jacobian regularization. In *International Conference on Machine Learning (ICML)*, 2021.

[5] S. Bai, Z. Geng, Y. Savani, and J. Z. Kolter. Deep equilibrium optical flow estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[6] S. Bai, V. Koltun, and J. Z. Kolter. Neural deep equilibrium solvers. In *International Conference on Learning Representations (ICLR)*, 2022.

[7] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018.

[8] T. Chen, J. B. Lasserre, V. Magron, and E. Pauwels. Semialgebraic representation of monotone deep equilibrium models and applications to certification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[9] L. El Ghaoui, F. Gu, B. Travacca, A. Askari, and A. Tsai. Implicit deep learning. *SIAM Journal on Mathematics of Data Science*, 3(3):930–958, 2021.

[10] 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.

[11] T. Gao, H. Liu, J. Liu, H. Rajan, and H. Gao. A global convergence theory for deep reLU implicit networks via over-parameterization. In *International Conference on Learning Representations (ICLR)*, 2022.

[12] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, and D. Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations (ICLR)*, 2019.

[13] S. Jafarpour, A. Davydov, A. Proskurnikov, and F. Bullo. Robust implicit networks via non-euclidean contractions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[14] K. Kawaguchi. On the theory of implicit deep learning: Global convergence with implicit layers. In *International Conference on Learning Representations (ICLR)*, 2021.

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

[16] C. K. Ling, F. Fang, and J. Z. Kolter. What game are we playing? end-to-end learning in normal and extensive form games. In *IJCAI'18*, 2018.

[17] C. Pabbaraju, E. Winston, and J. Z. Kolter. Estimating lipschitz constants of monotone deep equilibrium models. In *International Conference on Learning Representations (ICLR)*, 2021.

[18] Z. Ramzi, F. Mannel, S. Bai, J.-L. Starck, P. Ciuciu, and T. Moreau. SHINE: SHaring the INverse estimate from the forward pass for bi-level optimization and implicit models. In *International Conference on Learning Representations (ICLR)*, 2022.

[19] C. Rasmussen and Z. Ghahramani. Occam's razor. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

[20] M. Revay, R. Wang, and I. R. Manchester. Lipschitz bounded equilibrium networks. *ArXiv*, abs/2010.01732, 2020.

[21] Y. Rubanova, R. T. Q. Chen, and D. K. Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[22] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[23] H. Tanaka, D. Kunin, D. L. K. Yamins, and S. Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *International Conference on Neural Information Processing Systems*, 2020.

[24] C. Wang, G. Zhang, and R. Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020.

[25] P. Wang, P. L. Donti, B. Wilder, and J. Z. Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[26] C. Wei and J. Z. Kolter. Certified robustness for deep equilibrium models via interval bound propagation. In *International Conference on Learning Representations (ICLR)*, 2022.

[27] E. Winston and J. Z. Kolter. Monotone operator equilibrium networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[28] X. Xie, Q. Wang, Z. Ling, X. Li, Y. Wang, G. Liu, and Z. Lin. Optimization induced equilibrium networks. *ArXiv*, abs/2105.13228, 2021.

A Deep equilibrium models

In this section, we recall the definition of deep equilibrium models (DEQ). The reader is refered to [2, 9] for more details.

DEQ. Given a parametrized function f_θ (usually a shallow neural network), the forward pass of DEQ is defined by first solving the fixed point equation $z^\star = f_\theta(z^\star, x)$ for the feature $z^\star$, then passing the feature $z^\star$ through a classifier to obtain output. In the backward pass, one calculates the gradient of a loss function ℓ with respect to the weight θ by implicit differentiation

$$\frac{\partial \ell}{\partial \theta} = \frac{\partial \ell}{\partial z^\star}\Big(I - \frac{\partial f_\theta(z^\star, x)}{\partial z^\star}\Big)^{-1}\frac{\partial f_\theta(z^\star, x)}{\partial \theta}$$

for gradient descent [2, 9].

MonDEQ. In general, the existence and uniqueness of the fixed point $z^\star$ is not guaranteed. To ensure the well-posedness of the fixed point equation, some constraints on the parametrization of f_θ have been proposed in [27, 9]. In this paper, we have focused on the monotone deep equilibrium model (monDEQ) of [27] in which the well-posedness of DEQ layers is obtained by monotone operator conditions from operator splitting methods in convex optimization. In particular, for a DEQ layer $f_\theta(z, x) = \sigma(Wz + Ux + b)$, the parametrization $W = (1 - m)I - A^TA + B - B^T$ where $m > 0, A, B \in \mathbb{R}^{n\times n}$, ensures that the fixed point $z^\star = f_\theta(z^\star, x)$ exists and is unique for each input x. Operator splitting methods like forward-backward splitting or Peaceman-Rachford can be used to solve for $z^\star$ [27].

Fully connected monDEQ layer. On an input vector $x \in \mathbb{R}^d$, the fixed point feature $z \in \mathbb{R}^n$ is defined by a monDEQ layer with $W \in \mathbb{R}^{n\times n}$, $U \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, where we parameterize W directly by $A, B \in \mathbb{R}^{n\times n}$ in the monotone DEQ parametrization, and no constraint is put on U [27].

Convolutional monDEQ layer For a d-channel input $x \in \mathbb{R}^{ds^2}$ of size $s \times s$, the n-channel fixed point feature $z \in \mathbb{R}^{ns^2}$ is defined by a monDEQ with linear forms of 2D convolutions $W \in \mathbb{R}^{ns^2\times ns^2}$ and $U \in \mathbb{R}^{ns^2\times ds^2}$. Here, W is parameterized by two additional convolutions A, B of the same form [27].

B Neural network pruning

Given a neural network, we first devide the network into layers: $\ell = 0$ is the input layer, then for each layer $\ell \in \{1, 2, \dots, L\}$, we flatten the weights on the connections from layer $\ell - 1$ to layer ℓ into a weight vector $w_\ell \in \mathbb{R}^{d_\ell}$, where d_ℓ is the number of connections from layer $\ell - 1$ to layer ℓ. We write $W = (w_1, \dots, w_L)$.

Pruning, i.e. removing connections, is understood in terms of binary mask vectors $m_\ell \in \{0, 1\}^{d_\ell}$. After pruning, we obtain a subnetwork with weights $m_\ell \odot w_\ell$ - the elementwise product of masks and weights. Sparsity is defined as the fraction of weights being removed: $s = 1 - \frac{\sum m_\ell}{\sum d_\ell} \in [0, 1]$.

A pruning method usually consists of 2 operations: *score* and *remove*, where *score* takes as input weights of the network, and outputs an important score for each weights: $z_\ell = score(w_\ell) \in \mathbb{R}^\ell$; then *remove* takes as input the scores $(z_1, \dots, z_L)$ and the sparsity s, and outputs the masks m_ℓ with overall sparsity s.

Pruning can be done in one-shot or iteratively. For one-shot pruning, we only generate the scores once, then prune the network upto sparsity s. For iterative pruning, we repeat the processes of score, then prune from sparsity $s^{(n-1)/N}$ to sparsity $s^{n/N}$ iteratively N times.

We follow the following retraining scheme for comparison of performance of full and pruned networks. If we train the full network by t epochs, then we also train the subnetwork obtained after pruning by t epochs with the same learning rate schedule.

In this paper, we have performed experiments with pruning at initialization (PaI) methods for DEQs. A general references for pruning at initialization is [10]. We briefly recall the methods here. The most naive PaI method is random PaI (**Rand**) where one picks the weights to prune at uniformly

	MNIST		SHVH	CIFAR-10
	FC	Single conv	Single conv	Single conv
Num. channels	87*	54	81	81
Num. params	84.313	84.460	172.218	172.218
Epochs	40	40	40	40
Initial lr	0.001	0.001	0.001	0.001
Lr decay steps	10	10	25	10
Lr decay factor %	10	10	10	10

Table D1: Model hyperparameters. *FC is a dense layer with output dimension of 87.

random. Next, a strong baseline is magnitude PaI (**Mag**) where the weight scores are simply the absolute values of neural network weights; and one prunes the weights with smallest absolute values upto given sparsity.

SNIP. SNIP was introduced in [15] with the pruning objective of reducing connection sensitivity to the training loss. One passes a mini-batch of data through the network and compute the score s_i for each weight w_i of SNIP as $s_i = |w_i \frac{\partial \ell}{\partial w_i}|$.

GraSP. GraSP [24] is another gradient-based pruning PaI which aims to preserve the gradient flow of sparse networks obtained by pruning. The score of a weight w_i in GraSP is computed as $s_i = -w_i \cdot (Hessian \cdot Gradient)_i$, where $(Hessian \cdot Gradient)_i$ is the i-th component of the product between Hessian and gradient of the training loss after passing a mini-batch of data through the network.

Synflow. Synflow [23] is an iterative PaI. The pruning objective of Synflow is to make the network remains connected until the most extreme possible sparsity. The weight scores are computed as follows. One first replaces all weights in the network by their absolute values. Then, an all 1's input tensor is passed through the network, and one computes the sum of the logits as R. Finally, the score of a weight w_i is computed as $s_i = |w_i \cdot \frac{\partial R}{\partial |w_i|}|$. Synflow prunes the remaining weights with lowest scores from sparsity $s^{\frac{n-1}{N}}$ to sparsity $s^{\frac{n}{N}}$ iteratively N times for $n = 1, 2, \ldots, N$.

C Failure cases of pruning

- GraSP fails to calculate its score when using ReLU activation on the single convolution monDEQ models $\rightarrow$ We change the ReLU to the Sigmoid function when pruning, and return to ReLU when training the sparse monDEQ model.
- GraSP fails to calculate the Hessian matrix when pruning non-monotone DEQ, particularly resnet DEQ.

D Experiment details

We follow the setting of monDEQ [27] in both architecture and hyperparameters. The details are listed in the Table D1

D.1 Dense vs sparse experimental details

In the section, we describe the setting of dense models which have the same number of parameters as the corresponding sparse models. We follow Appendix M of [10] to compute the number of effective parameters. Table D2 list the number of parameters in pruned models and the number of effective parameters of models pruned by SNIP and Magnitude. We also list the number of channels in dense models such that the number of parameters in dense models is similar to pruned models.

With the same sparsity ratio, the higher the number of effective parameters, the higher density in the representation of z, since sparse z leads C matrix in classifier has many redundant columns. Table D2 shows that SNIP produces subnetworks with much more effective parameters than Magnitude.

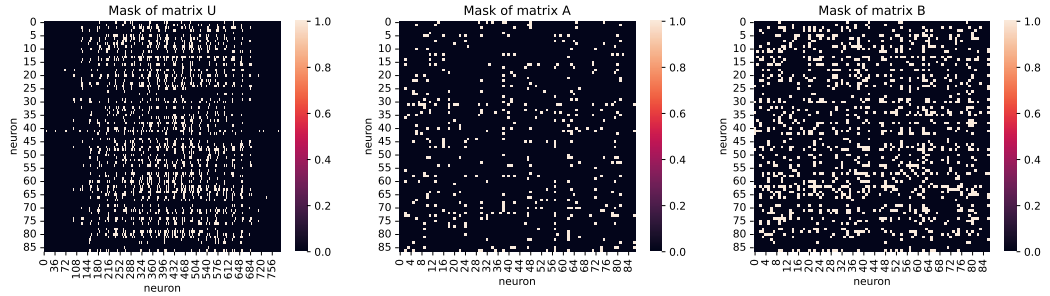
Sparsity	10	50	70	90	95	97	99	99.9
Num. channels	81	81	81	81	81	81	81	81
Num. remaining params	160.190	112.076	88.019	63.962	57.948	55.542	53.136	52.054
SNIP								
Num. effective params	159.532	111.475	87.413	63.361	57.369	54.908	52.651	52.027
Num. channels*	77	62	54	44	41	40	39	38
Num. params*	158.168	110.618	88.570	64.250	57.656	55.530	53.440	51.386
Magnitude								
Num. effective params	134.209	86.619	63.182	38.103	32.823	29.931	28.030	27.138
Num. channels*	70	53	44	31	28	26	25	24
Num. params*	134.970	85.976	64.250	38.016	32.826	29.546	27.960	26.410

Table D2: Detailed setting of dense single convolution architecture on CIFAR-10. * indicates the number of channels and parameters of the dense models.

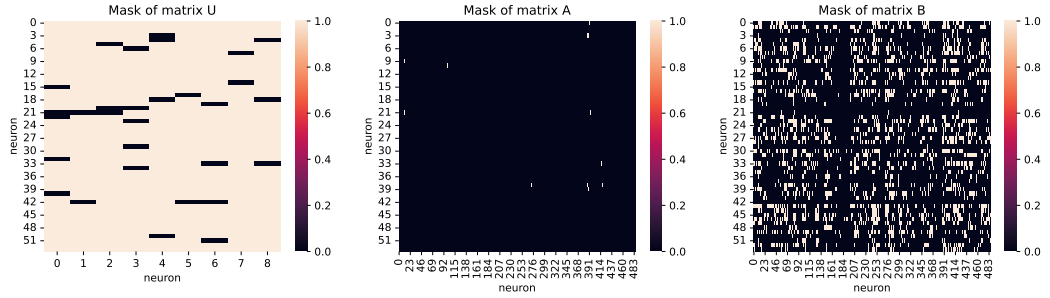
D.2 Sparsity patterns in weight matrices of monDEQ models

Here, we visualize the mask of each weight matrix in the monDEQ models pruned 90% by SNIP. With convolution layer, the default weight matrix has the format of $W \in \mathbb{R}^{C_{out} \times C_{in} \times H \times W}$, in which $C_{out}\, C_{in}$ are the number of channels in output, input feature maps, H , W are the height and width of each filter. To be better visualization, we reshape the convolution weight to $W' \in \mathbb{R}^{C_{out} \times C_{in} * H * W}$. Figure D1 illustrates masks of weight matrices on the different monDEQ models and datasets. The x-axis and y-axis are the column and row of the weight matrix respectively.

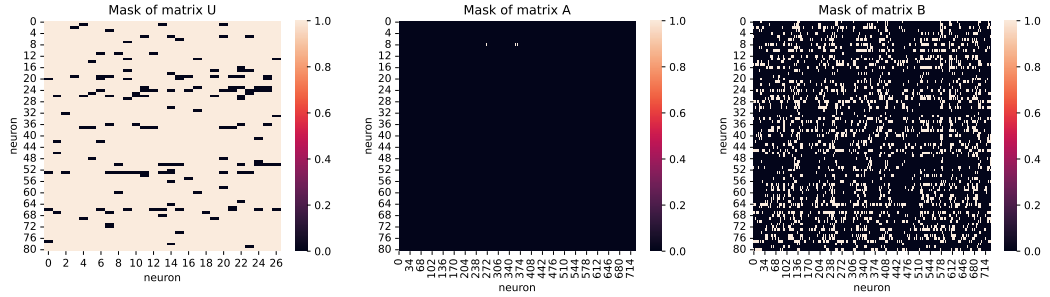
It is easy to see that except for FC monDEQ on MNIST, all models are pruned clearly in A matrices, while U matrices are almost intact. The B matrices are pruned in a large amount.



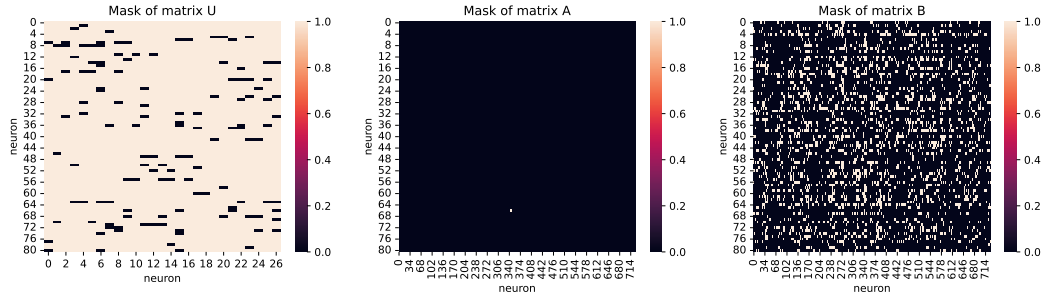
(a) The sparsity patterns of weight matrices in FC monDEQ on MNIST



(b) The sparsity patterns of weight matrices in single conv monDEQ on MNIST



(c) The sparsity patterns of weight matrices in single conv monDEQ on SHVN



(d) The sparsity patterns of weight matrices in single conv monDEQ on CIFAR-10

Figure D1: Mask visualizations of weight matrices in 90% pruned monDEQ models by SNIP on different datasets.