Out-of-Distribution Generalization with Deep Equilibrium Models

Kaiqu Liang [*1 2] **Cem Anil** [*1 2] **Yuhuai Wu** [1 2] **Roger Grosse** [1 2]

Abstract

Deep learning models often make unexpected mistakes under distribution shifts, preventing their widespread adoption in safety-critical applications. In this paper, we investigate whether Deep Equilibrium (DEQ) Models generalize better under systematic distribution shifts than their fixed-depth counterparts. We present two sets of experiments to address this question, both of which indicate that DEQ models enjoy superior out-of-distribution generalization. We first observe, on various tasks, that DEQ models spend more time processing inputs of greater complexity, in a trend that extends predictably to levels of complexity larger than those observed during training. We then inspect how the internal representations of DEQ models derived from out-of-distribution (OOD) samples change as they approach equilibria. We find that the statistics of the internal representations of OOD samples are drawn closer to those derived from in-distribution samples in DEQ models, in sharp contrast to the behavior of fixed-depth architectures. Based on these results, we hypothesize that the convergence-based forward-pass termination criterion of DEQ models endows them with an inductive bias towards better out-of-distribution generalization.

1. Introduction

Despite their strong performance on a large variety of applications, deep learning models often display erratic, unpredictable behaviour when tested under distribution shifts and can be confident of wrong predictions. (Che et al., 2019; Sastry & Oore, 2020) In this work, we investigate whether Deep Equilibrium Models (Bai et al., 2019), a class of attractor neural networks, have better out-of-distribution generalization properties than their standard, fixed-depth counterparts. Instead of explicitly representing a (parametrized) mapping, attractor neural networks implicitly define a mapping from inputs to internal representations in the form of a fixed-point equation. DEQ models explicitly solve for the fixed-point using root finders.

We present empirical evidence that suggests DEQ models are more robust against distribution shifts. We establish that, on various tasks aimed at testing different notions of OOD generalization, DEQ models indeed perform better than their fixed-depth counterparts. To shed light on this, we present two addition sets of results: 1) DEQ models spend more iterations processing inputs of greater complexity, in a trend that extends predictably to levels of complexity larger than those observed during training. 2) Each application of the DEQ root finding operation draws the statistics of the internal representations of OOD samples closer to those derived from in-distribution samples in DEQ models. This is unlike what happens in fixed-depth networks where OOD inputs result in features whose statistics differ significantly from those obtained from in-distribution inputs.

Based on these results, we hypothesize that the convergence-based forward-pass termination criterion of DEQ models endows them with an inductive bias towards better out-of-distribution adaptation. We believe implicit models present a promising avenue towards building architectures that have better OOD generalization capabilities.

2. Background

Attractor Networks: Attractor networks are a class of neural network model that treat computing internal representations as a fixed-point finding problem. (McClelland & Rumelhart, 1989) Concretely, letting $\mathbf{x} \in \mathbb{R}^{n_x}$ and $f_{\mathbf{w}} : \mathbb{R}^{n_x \times n_z} \mapsto \mathbb{R}^{n_z}$ stand for an input and the attractor network function (or "cell") parametrized by $\mathbf{w} \in \mathbb{R}^{n_w}$ respectively, attractor networks aim to solve for the fixed point $\mathbf{z}^* \in \mathbb{R}^{n_z}$ that satisfies $\mathbf{z}^* = f_{\mathbf{w}}(\mathbf{x}, \mathbf{z}^*)$. The cell $f_{\mathbf{w}}$ might represent anything from a fully connected layer to a transformer block (Vaswani et al., 2017).

The most straightforward approach to solve for fixed-points is the *fixed point iteration* method, which recursively applies the function $f_{\mathbf{w}}$ on the internal representations $\mathbf{z}$ (i.e. $\mathbf{z}_{t+1} = f_{\mathbf{w}}(\mathbf{x}, \mathbf{z}_t)$) until it converges to a fixed point:

[*]Equal contribution [1]Department of Computer Science, University of Toronto [2]Vector Institute. Correspondence to: Cem anil <anilcem@cs.toronto.edu>, Kaiqu Liang <kaiqu.liang@mail.utoronto.ca>.

Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning.,

$\mathbf{z}^* = f_{\mathbf{w}}(\mathbf{x}, \mathbf{z}^*)$. Since solving for fixed points exactly is costly, fixed point iterations are often terminated when the norm of the difference between subsequent iterates falls below a pre-selected threshold.

Deep Equilibrium (DEQ) Models: Deep equilibrium models (Bai et al., 2019) are a type of attractor network model that solves for fixed points explicitly using black-box root finders, such as Broyden's method or Anderson accelaration (Bai et al., 2019; Kolter et al., 2020). In order to avoid differentiating through the root-finding procedure, DEQ models utilize implicit differentiation to compute gradients. Concretely, letting $g_{\mathbf{w}}(\mathbf{x}, \mathbf{z}^*) = f_{\mathbf{w}}(\mathbf{x}, \mathbf{z}^*) - \mathbf{z}^*$ for a fixed point $\mathbf{z}^*$, the Jacobian of $\mathbf{z}^*$ with respect to the DEQ model weights can be given by:

$$\frac{d\mathbf{z}^*}{d\mathbf{w}} = -\left(\frac{\partial g_{\mathbf{w}}(\mathbf{x}, \mathbf{z}^*)}{\partial \mathbf{z}^*}\right)^{-1} \frac{\partial f_{\mathbf{w}}(\mathbf{x}, \mathbf{z}^*)}{\partial \mathbf{w}}$$

DEQ models have been shown to achieve state-of-the-art performance on various tasks including language modelling and image classification. (Bai et al., 2019; 2020)

3. Evaluating OOD-Generalization of DEQ Models

We consider the following three synthetic tasks to evaluate different aspects of OOD generalization:

Blurry MNIST Task: The Blurry MNIST dataset is generated by convolving MNIST images by Gaussian filters of varying standard deviations. We trained on a uniform mixture of standard deviations 2, 2.5, 3 and 3.5. We generate 8 distinct test sets by varying the standard deviation from 2 up to 5.5 in increments of 0.5. The first 4 test sets were used to evaluate in-distribution generalization, while the remainder were used to evaluate OOD generalization.

Sequence Copy Task: This is the sequence modelling problem of producing an output sequence identical to a given source sequence. We trained on sequences of length 10, 20, 30 and 40, and tested, separately, on lengths 10 to 80 (in increments of 10).

Symbolic Rewrite Task: The symbolic rewrite task, proposed by Wu et al. (2021), is a sequence task that involves rewriting a base sequence using a given transformation rule. The following is an example:

source: $a + b - c \ \langle s \rangle \ A + B = B + A$
target: $b + a - c$

Here, the target rule $A + B = B + A$ is used to rewrite the $a + b$ pattern in the source sequence as $b + a$ to obtain the output sequence $b + a - c$. During training, we varied the number of rewrite rules between 1 and 4. While testing, we used between 1 to 8 rewrite rules to test both in and out-of-distribution generalization. More information about this task can be found in Appendix A.

The aforementioned tasks allow us to test in and out-of-distribution generalization performance by systematically applying varying degrees of distribution shift.

3.1. Models, Baselines and Training Details

For the Blurry MNIST task, we used a one layer fully connected DEQ cell. For the Sequence Copy and Symbolic Rewrite tasks, we used a transformer-based cell, similar to that used by Bai et al. (2019). We provide further details about the models in Appendix A. For each of the tasks described above, we trained fixed-depth baseline models. We tied the weights across different layers and used the same parametrization as in the corresponding DEQ models to ensure they have the same number of parameters. We fixed the depth roughly at the number of root finding operations the corresponding DEQ models take to converge.

3.2. OOD Generalization Results

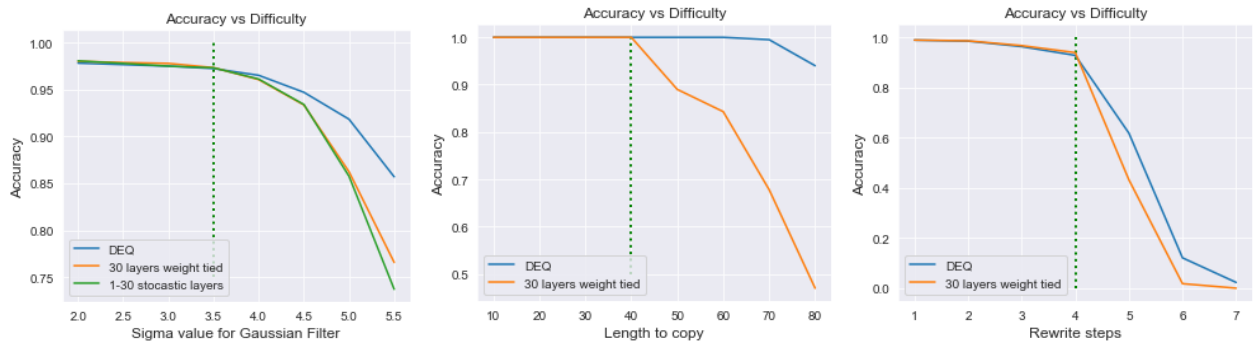
Figure 1 shows the in-distribution and out-of-distribution generalization performance (quantified by accuracy) of both the DEQ models and their fixed-depth counterparts. Both models achieve virtually identical in-distribution generalization performance. On OOD samples, however, the performance of the fixed-depth models degrades much more sharply than that of the DEQ models. This is especially prominent in the Sequence Copy task, where the DEQ models' performance remains perfect for a wide range of sequence lengths not seen during training.

As we elaborate further in Section 4, it takes a different number of root-finding iterations for DEQ models to converge for different inputs. To see if this variable-depth property alone could reproduce the OOD generalization behaviour of DEQ models, we also trained a weight-tied (non-DEQ) model where we varied the depth of the model stochastically. On the Blurry MNIST task, this in fact lead to a slight decrease in OOD generalization performance.

4. Adaptive Computation Behaviour

DEQ models have the flexibility to "vary their depth" on different inputs. This adaptive computation time capability could be useful for many OOD generalization tasks — especially in tasks where the network might be given inputs of varying complexity that are possibly unseen during training.

On the three tasks defined in Section 3, we tested whether DEQ models did indeed take more time to converge on examples of larger complexity/ambiguity. On the Blurry MNIST, Sequence Copy and Symbolic Rewrite tasks, we used the level of blur, length of sequence and number of



(a) Blurry MNIST task (b) Sequence Copy task (c) Symbolic Rewrite task

Figure 1. **Generalization results:** In-distribution and OOD generalization performance of DEQ models and their fixed-depth counterparts, on the (a) Blurry MNIST, (b) Sequence Copy and (c) Symbolic Rewrite tasks. On all tasks, while the in-distribution performance (left of the vertical line) is virtually identical, the DEQ model performs better on OOD examples.

rewrite rules to gauge the "complexity" of processing inputs, respectively. We computed, for each level of complexity, the average number of root finding operations the DEQ models take before converging. The results are shown in Figure 2. The average number of fixed-point iterations do indeed correlate strongly with the complexity of the inputs.

We also trained a non-DEQ, weight-tied model where we varied the depth of the model stochastically, in order to check if the variable-depth property of DEQ models could explain their strong generalization behaviour. During training, we sampled the depth uniformly between one and the depth of the fixed-depth model. During test time, we kept the depth constant. We found that the stochastic depth model achieved slightly worse OOD generalization performance than the fixed-depth one on the Blurry MNIST task.

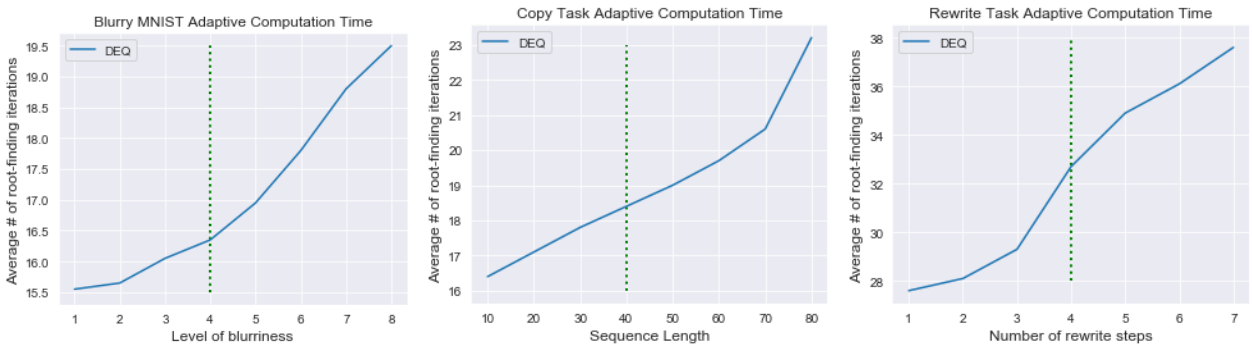
5. Evolution of OOD Representations in DEQ Models

We inspected how the statistics of the DEQ internal representations derived from OOD samples differ from those derived from in-distribution ones as they approached equilibria. We used the Gram-OOD method proposed by Sastry & Oore (2020) to quantify how much these statistics deviated from one another. We picked the Gram-OOD method as this approach has been used to achieve state-of-the-art OOD detection results on a variety of domains (Sastry & Oore, 2020) by only using in and out-of-distribution activation statistics. The Gram-OOD method works by computing elementwise min and max values of the Gram matrices computed using layer-wise activations obtained from the training set. These are then compared these with those obtained from a test example to compute a scalar *deviation score*, which quantifies how "out-of-distribution" the given test example is. We describe the full Gram-OOD method in Appendix B.

We adjusted the Gram-OOD method slightly when applying it to DEQ models, as DEQ models don't have "layers" in the traditional sense. We did this by computing the in-distribution min/max statistics by using all of the root-finding iterates encountered during the forward pass of DEQ models (i.e. by treating all iterates as if they belong to the same layer). When given a test example (both in-distribution and out-of-distribution), we computed how the deviations changed with each application of the root-finding algorithm to check if there is indeed a downward trend. We ran the same analysis on the fixed-depth baseline models to see if there's a qualitative difference.

We display our results on how the layer-wise deviations evolved with depth for both the DEQ and baseline models on the Blurry MNIST task in Figure 3. Figure 3a shows the results for in-distribution test examples. For both DEQ models and the baselines, the deviations decreased substantially before fluctuating around a constant value. This is unlike the results we obtained when we used OOD test samples (with blur level 6) (Figure 3b). While the same downward trend held for the DEQ model, the deviations actually increased considerably for fixed-depth baselines. These results imply that the fixed points derived by DEQ models from OOD inputs are less likely to be detected as "out-of-distribution" by the Gram-OOD test as they approach equilibrium. This is unlike what happens with fixed-depth networks, where hidden features of OOD inputs are consistently detected as out-of-distribution throughout the depth of the network. The evolution of the deviations for other blur levels can be found in Appendix C.

A major difference between DEQ and their fixed models is that the DEQ models' forward pass terminates iff the internal representations reach a fixed point. This constrains the internal representations to lie on the manifold defined by the fixed points of the DEQ cell. Consider the following possibility: *DEQ models don't have spurious equilibria: All fixed-points of a learned DEQ mapping correspond to*



(a) Blurry MNIST task (b) Sequence Copy task (c) Symbolic Rewrite task

Figure 2. **Adaptive Computation Time:** DEQ models naturally take more iterations to find fixed-points on inputs of higher complexity.

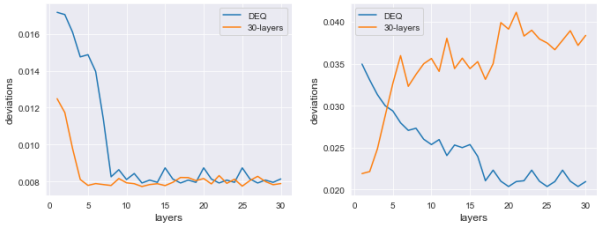
internal representations of the inputs that are useful for classification. If this is (even partially) true, it could help explain why DEQ models seem to enjoy superior OOD generalization performance. This conjecture suggests that when a DEQ model processes an OOD input, its internal representations will either diverge (in which case no output can be produced), or be pulled towards the manifold of equilibria encountered during training. Note that there exist encouraging results in literature about the lack of spurious equilibria in attractor networks. (Radhakrishnan et al., 2020)

To check the prevalence of spurious equilibria in DEQ models, we tested if there are OOD inputs which do converge to a fixed point, but still have significant deviation. On the Blurry MNIST task (using blur level 5), we found that only 1.2% of all fixed-point deviations were outside of 0.1 standard deviation of the population mean. Excluding these outliers lead to a mean of 0.0058, which is comparable to the average in-distribution deviation. This is in line the hypothesis that DEQ models have (almost) no spurious equilibria.

Our results based on Gram-OOD deviations provide indirect evidence for the no-spurious-equilibria hypothesis. Further research is needed to gain a full understanding of the aforementioned phenomena and rule out other competing hypotheses. It is also important to investigate under what the conditions spurious equilibria can arise and disappear.

6. Related Work

Dehghani et al. (2018) show that Universal Transformers, a variable-depth transformer architecture, can learn functions that have favourable OOD-generalization properties. The fixed-depth, weight tied baseline models we trained for the Sequence Copy and Symbolic Rewrite tasks strongly resemble the Universal Transformer architecture. Kaiser & Sutskever (2015) also develop methods to train models that generalize to OOD inputs by training their proposed Neural GPU architecture to explicitly represent algorithms. Their approach requires the use of a curriculum. Graves (2016) propose an approach to allow recurrent neural networks to spend varying amounts of time on different outputs. Winston & Kolter (2020) propose training DEQ models with a known Lipschitz constant to make them more robust against adversarial attacks.



(a) In distribution deviations (b) OOD deviations

Figure 3. **Evolution of deviations with depth on the Blurry MNIST task:** (a) For in-distribution test examples, the statistics of representations differ (quantified by Gram-OOD deviations on the y axis) less and less from those of the training examples with increasing depth. (b) For OOD test examples, the deviations only decrease for the DEQ model. This implies that the internal representations of DEQ models derived from OOD examples are pulled towards the manifold of fixed points encountered during training. No such mechanism exists for fixed-depth networks.

7. Conclusion

We presented experimental evidence suggesting that DEQ models enjoy superior out-of-distribution generalization compared to their fixed-depth counterparts. To explain this finding, we ran further experiments that showed: 1) DEQ models take more time to find fixed points corresponding to inputs of greater complexity. 2) The statistics of the internal representations of OOD samples are drawn closer to those derived from in-distribution samples in DEQ models, in sharp contrast to the behavior of fixed-depth architectures. Our findings motivate the study of attractor models in the context of building robust models that generalize more seamlessly under distribution shifts than existing architectures.

Acknowledgements

Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. We thank Chandramouli Sastry for his support on running the Gram-OOD (Sastry & Oore, 2020) evaluations.

References

Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. *arXiv preprint arXiv:1909.01377*, 2019.

Bai, S., Koltun, V., and Kolter, J. Z. Multiscale deep equilibrium models. *arXiv preprint arXiv:2006.08656*, 2020.

Che, T., Liu, X., Li, S., Ge, Y., Zhang, R., Xiong, C., and Bengio, Y. Deep verifier networks: Verification of deep discriminative models with deep generative models. *arXiv preprint arXiv:1911.07421*, 1, 2019.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, Ł. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Graves, A. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Kaiser, Ł. and Sutskever, I. Neural gpus learn algorithms. *arXiv preprint arXiv:1511.08228*, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kolter, Z., Duvenaud, D., and Johnson, M. Tutorial: Deep implicit layers-neural odes, deep equilibirum models, and beyond, 2020.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

McClelland, J. L. and Rumelhart, D. E. *Explorations in parallel distributed processing: A handbook of models, programs, and exercises*. MIT press, 1989.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Radhakrishnan, A., Belkin, M., and Uhler, C. Overparameterized neural networks implement associative memory. *Proceedings of the National Academy of Sciences*, 117 (44):27162–27170, 2020.

Sastry, C. S. and Oore, S. Detecting out-of-distribution examples with gram matrices. In *International Conference on Machine Learning*, pp. 8491–8501. PMLR, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Walker, H. F. and Ni, P. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

Winston, E. and Kolter, J. Z. Monotone operator equilibrium networks. *arXiv preprint arXiv:2006.08591*, 2020.

Wu, Y., Rabe, M., Li, W., Ba, J., Grosse, R., and Szegedy, C. Lime: Learning inductive bias for primitives of mathematical reasoning. *arXiv preprint arXiv:2101.06223*, 2021.

A. Dataset, Model and Training Details

A.1. Details of the Symbolic Rewrite Task

The symbols used to generate the sequence to be rewritten (source) and the rewrite rules are different: We use lowercase letters (e.g. a, b, c,...) and standard math operators (e.g., "$+ - * = ()\&$") to construct the source sequences. We use capital letters (e.g., A, B, C...) and math operators to specify the rewrite rules. The difficulty of the problem instances can be increased by increasing number of rewrite rules.

In our experiments, we use 40 distinct symbols to construct the source sequences, and 6 upper case symbols to construct the target rules. We varied the length of the pattern to be rewritten between 3 and 5 and varied the number of rewrite rules between 1 and 7.

A.2. Blurry MNIST Training Details

Model: We parameterized the DEQ cell using a single-layer, 90 units wide fully connected layer with a ReLU activation. Instead of concatenating the input and the fixed point, we added them before feeding to the DEQ cell. We linearly transformed the inputs to have a dimensionality of 90 so that they could be directly summed with the fixed points. This way of fusing the input and fixed point was also used by Bai et al. (2019) in their language modelling experiments.

For solving the fixed points during forward pass, we used the Anderson root finder (Walker & Ni, 2011) with a memory size of 5, setting the λ and β hyperparameters to be $1e-4$ and 1, respectively. This choice of hyperparameters was also used by Kolter et al. (2020). We terminated the solver when the L2 norm of the difference between the subsequent iterates fell below the threshold $1e-3$, or the number of iterations surpassed 30.

Optimizer: We used the Adam optimizer (Kingma & Ba, 2014) with a learning rate of $1e-3$. We varied the learning rate with a step schedule with a step size of 50 (in epochs) and decay factor of 0.1. We used a batch size of 500.

We solved for the fixed-point equation that arises when computing the implicit gradients using the Anderson Solver with the same hyperparameters that we used for forward pass. We set the termination threshold to $4.2e-8$.

Regularization: We applied weight decay with a coefficient of $1e-3$.

A.3. Sequence Copy Training Details

Model: We parameterized the DEQ cell using the transformer decoder block from the GPT2 architecture (Radford et al., 2019) We set the embedding dimensionality to 384, number of heads to 6 and maximum sequence length to 180. We did not apply any dropout through the model. For solving the fixed points during forward pass, we used the Anderson root finder (Walker & Ni, 2011) with a memory size of 5, setting the λ and β hyperparameters to be $1e-4$ and 1, respectively.

Data augmentation: Similar to what Dehghani et al. (2018) do in their Sequence Copy experiments, we start the source sequence with randomized offsets. This allows the transformer to learn position-relative transformations and makes it possible to learn to to copy longer sequences from a dataset of shorter sequences.

Optimizer: We used the AdamW optimizer (Loshchilov & Hutter, 2017) with a learning rate of $1e-4$. We varied the learning rate with a cosine annealing schedule (Loshchilov & Hutter, 2016). We used a batch size of 60. We solved for the fixed-point equation that arises when computing the implicit gradients using the Anderson Solver with the same hyperparameters that we used for forward pass. We set the termination threshold to $3e-8$. When computing the prediction loss, we masked out the contributions from predicting the tokens in the source sequence. We trained the model for 200000 gradient updates.

Regularization: We applied weight decay with a coefficient of $1e-3$. We found that it was helpful to penalize the L2 norm of the fixed points to speed up convergence. We added the regularization term $\lambda||\mathbf{z}||_2^2$ and set λ to be 0.2.

Pretraining: Similar to the results of Bai et al. (2019) on language modelling, we found pretraining to be quite useful for stabilizing DEQ training. Following their procedure, we pretrained the DEQ model, treating it as a six-layer, weight tied model, for $200k$ gradient steps.

Tuning: We did a grid search on the learning rate ($\{10^{-4}, 1.5 \cdot 10^{-4}, 2.5 \cdot 10^{-4}\}$), weight decay coefficient ($\{0, 0.001\}$) and the fixed-point L2 penalty coefficient (λ) $\{0, 0.1, 0.2, 0.5\}$) before settling on the final values.

A.4. Symbolic Rewrite Training Details

Model, Optimizer and Regularization: We used the same model we used for the Sequence Copy task, with the exception that the sequence length was set to 160. We used the same optimizer setup we used for the Sequence Copy task. We applied weight decay with a coefficient of $1e-3$ and set the fixed-point L2 penalty coefficient to be 0.2.

Pretraining: we pretrained the DEQ model, treating it as a two layer, weight tied model, for 20000 gradient steps.

Tuning: We did a grid search on the learning rate ($\{10^{-4}, 1.5 \cdot 10^{-4}, 2.5 \cdot 10^{-4}\}$), weight decay coefficient ($\{0, 0.001\}$) and the fixed-point L2 penalty coefficient (λ) $\{0, 0.1, 0.2, 0.5\}$) before settling on the final values.

B. Outlier Detection using Gram-OOD

Algorithm box 1 describes how the Gram-OOD method computes the min and max statistic that are used to compute deviations. The p-th order Gram matrices are computed by $G^p = (F^p F^{p^T})^{\frac{1}{p}}$ where F stand for an activation vector.

Element-wise deviations are computed by comparing, element-by-element, the values of the Gram matrices obtained using a test example with the min/max statistics obtained from the training set, using the following equation:

$$\delta(\text{min}, \text{max}, \text{value}) = \begin{cases} 0 & \text{min} < \text{value} < \text{max} \\ \frac{\text{min}-\text{value}}{|\text{min}|} & \text{value} < \text{min} \\ \frac{\text{value}-\text{max}}{|\text{max}|} & \text{value} > \text{max} \end{cases}$$

The total layer-wise deviation scores are finally computed by summing all elementwise deviations corresponding to all Gram matrix orders.

Algorithm 1 Compute the minimum and maximum values of feature co-occurrences for each class, layer and order

Input:
C: Number of output classes
L: Number of Layers in entire network
P: Set of all orders of Gram Matrix to consider
$Train$: The train data
Output:
Mins, Maxs

Mins[C][L][P] $\left[\max_{1 \le l \le L} \frac{n_l(n_l+1)}{2}\right] \leftarrow \infty$
Maxs[C][L][P] $\left[\max_{1 \le l \le L} \frac{n_l(n_l+1)}{2}\right] \leftarrow -\infty$
for c in $[1, C]$ **do**
  $Train_c = \{D \mid D \in Train \; s.t. \; f(D) = c\}$
  for D $\in Train_c$ **do**
    for l in $[1, L]$ **do**
      for p in P **do**
        stat = $\overline{G_l^p(D)}$
        for i in $\left[1, \frac{n_l(n_l+1)}{2}\right]$ **do**
          Mins$[c][l][p][i]$ = min(Mins$[c][l][p][i]$,stat$[i]$)
          Maxs$[c][l][p][i]$ = max(Maxs$[c][l][p][i]$,stat$[i]$)
return Mins, Maxs

C. Evolution of Deviations with Depth

Figure 4 displays how deviations evolve with depth for (OOD) blur levels 5, 6, 7 and 8.

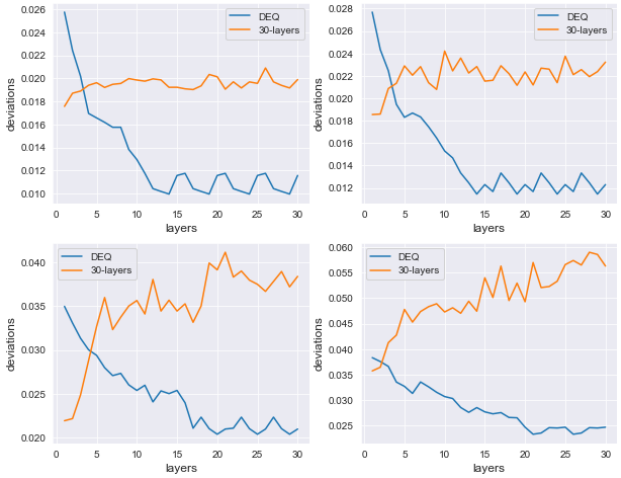


Figure 4. Blue curve shows the layer-wise deviations of the DEQ Model on the Blurry MNIST task. Orange curve is layer-wise deviation of 30 layers weight tied network. *Top Left:* Layer-wise deviation computed by examples with blur level 5. *Top Right:* 6. *Bottom Left:* 7. *Bottom Right:* 8.