Deep Equilibrium Models

Troshin Sergey

Higher School of Economics

https://arxiv.org/pdf/1909.01377.pdf

Deep Equilibrium Models

Shaojie Bai
Carnegie Mellon University

J. Zico Kolter
Carnegie Mellon University
Bosch Center for AI

Vladlen Koltun
Intel Labs

- Advances in Neural Information Processing Systems (NeurIPS), 2019 (Selected for spotlight oral presentation)

Outline

- Introduction
- Deep Equilibrium Models
- Experiments
- Properties of the DEQ
- Universality

Sequence modeling task

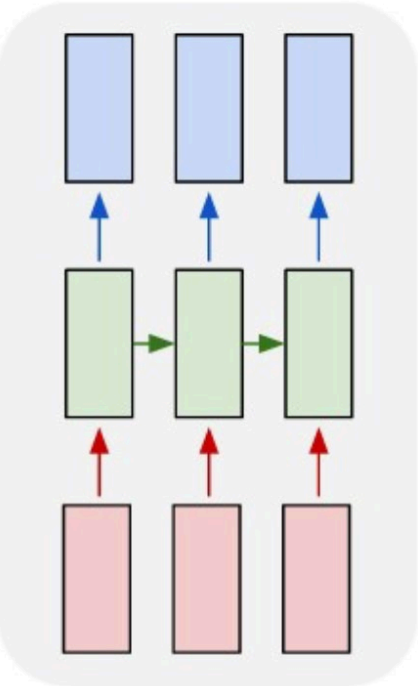
$\mathbf{x}_{1:T} = [x_1, \ldots, x_T] \in \mathbb{R}^{T \times p}$ where $x_i \in \mathbb{R}^p$ – inputs of the model

$G(\mathbf{x}_{1:T}) = \mathbf{y}_{1:T} \in \mathbb{R}^{T \times q}$ – outputs of the model

Constraint: causality

Applications:

- Language modeling
- Time series tasks



Limitations of using very deep neural networks.

- Need O(L) memory for training, L – the number of layers.

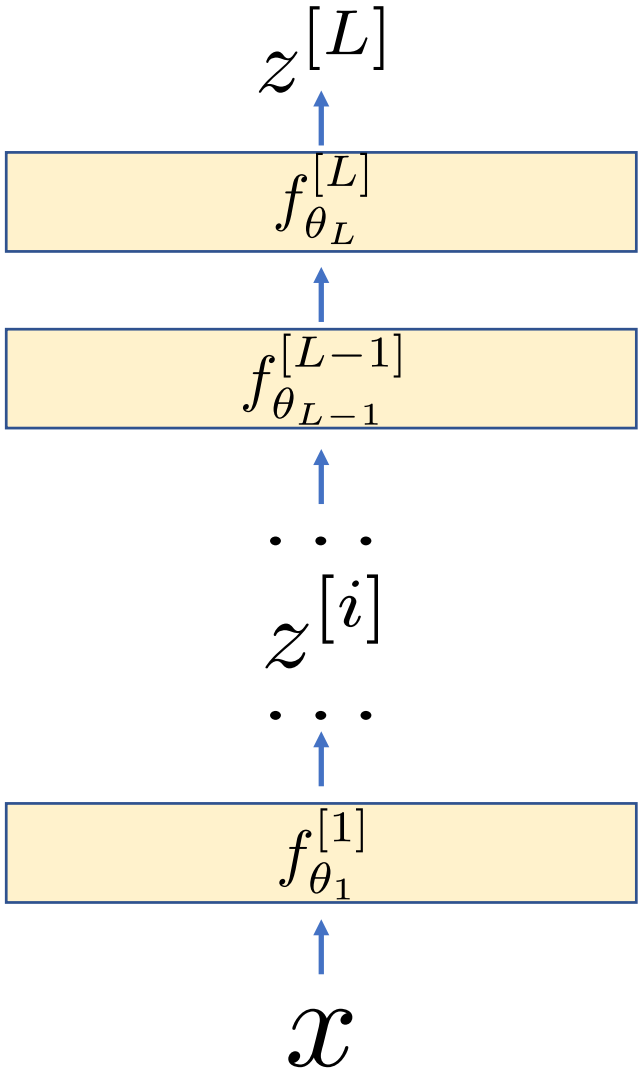
Solutions:

- Gradient Checkpointing: $\sqrt{L}$ tradeoff for some extra forward passes
- Neural ODEs: Constant (using black-box solver for backward pass)

https://github.com/cybertronai/gradient-checkpointing

https://arxiv.org/abs/1806.07366

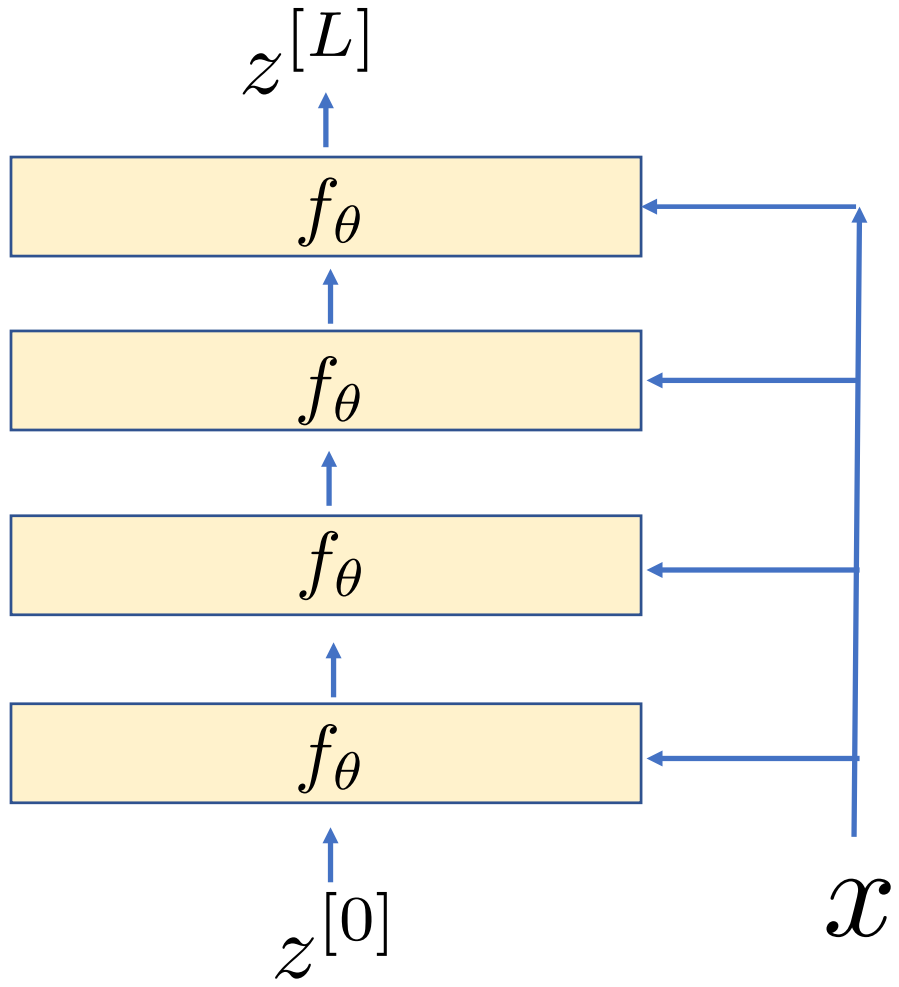
Common deep sequence model



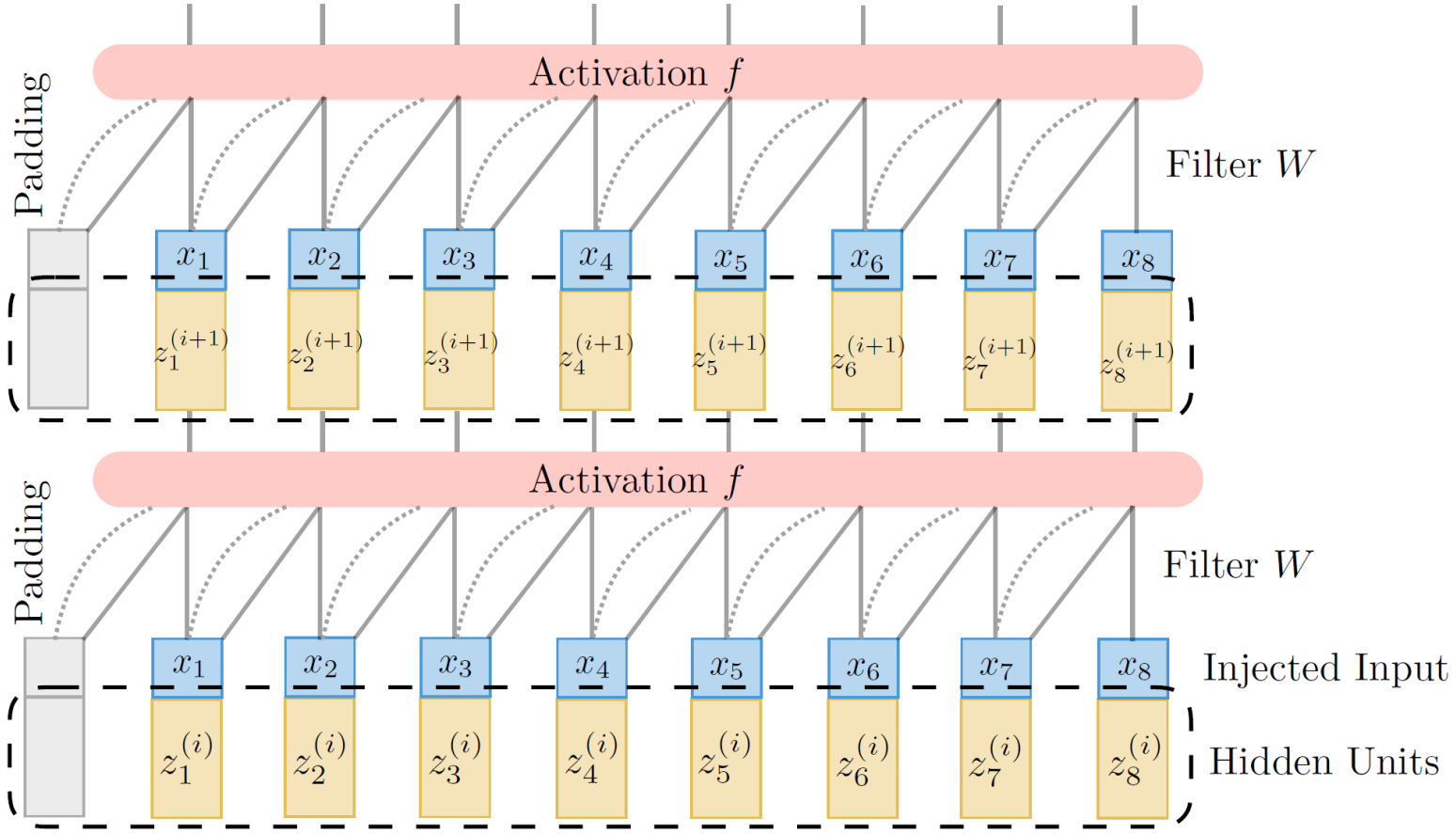
Weight-tied Input-Injected DNN

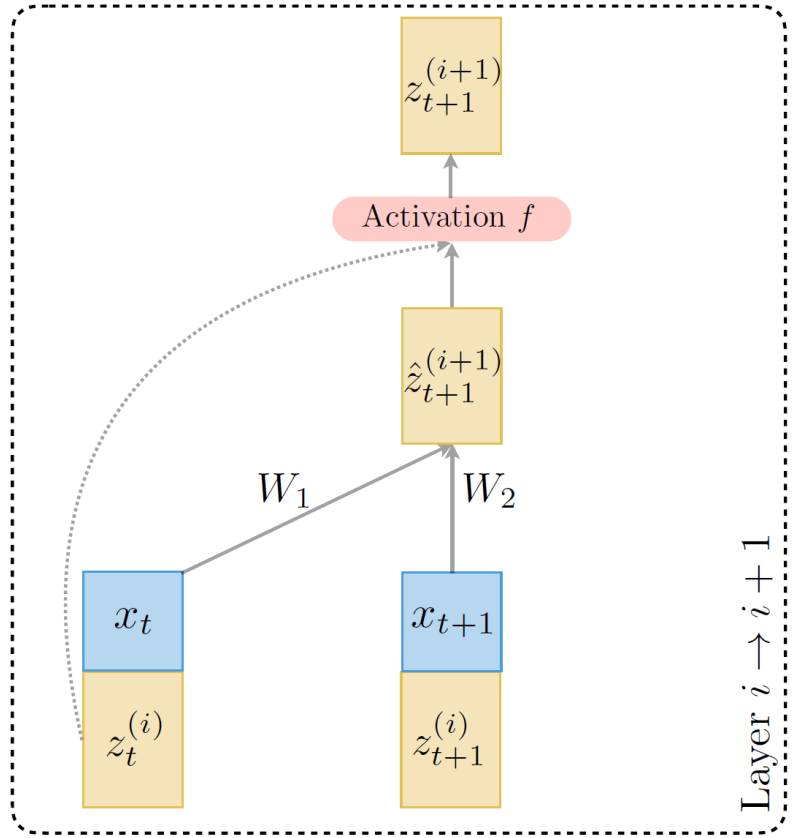
$$\mathbf{z}_{1:T}^{[i+1]} = f_\theta(\mathbf{z}_{1:T}^{[i]}, \mathbf{x}_{1:T}),\ i = 1, \ldots, L-1$$

$$\mathbf{z}_{1:T}^{[0]} = \mathbf{0}, \quad G(\mathbf{x}_{1:T}) \equiv \mathbf{z}_{1:T}^{[L]}$$



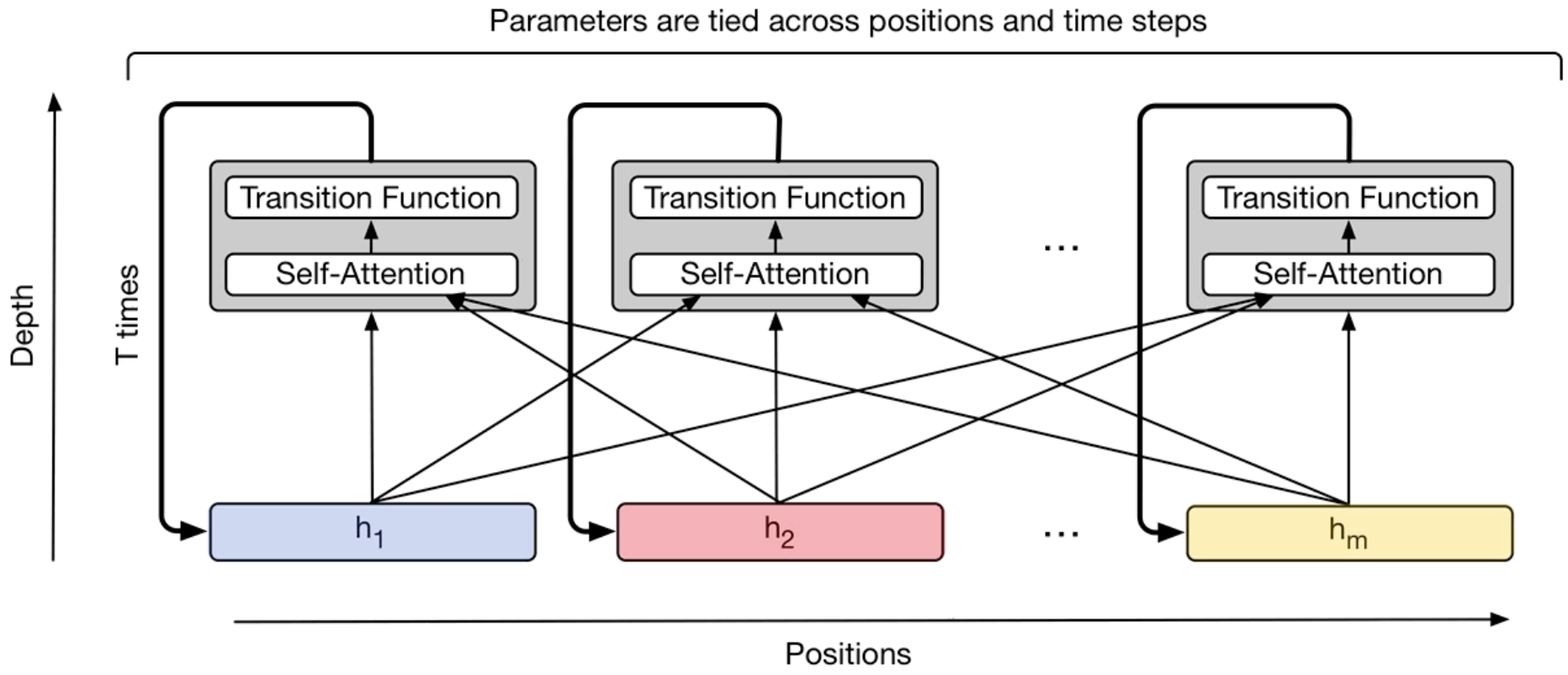
Trellis network (2019)





$$f_\theta(\mathbf{z}_{1:T}; \mathbf{x}_{1:T}) = \psi(\text{Conv1D}([\mathbf{u}_{-(k-1)s:}, \mathbf{z}_{1:T}]; W_z) + \tilde{\mathbf{x}}_{1:T})$$

Universal Transformer (2019)



$$f_\theta(\mathbf{z}_{1:T}; \mathbf{x}_{1:T}) = \mathsf{LN}(\phi(\mathsf{LN}(\mathsf{SelfAttention}(\mathbf{z}_{1:T}W_{QKV} + \tilde{\mathbf{x}}_{1:T}; \mathbf{PE}_{1:T}))))$$

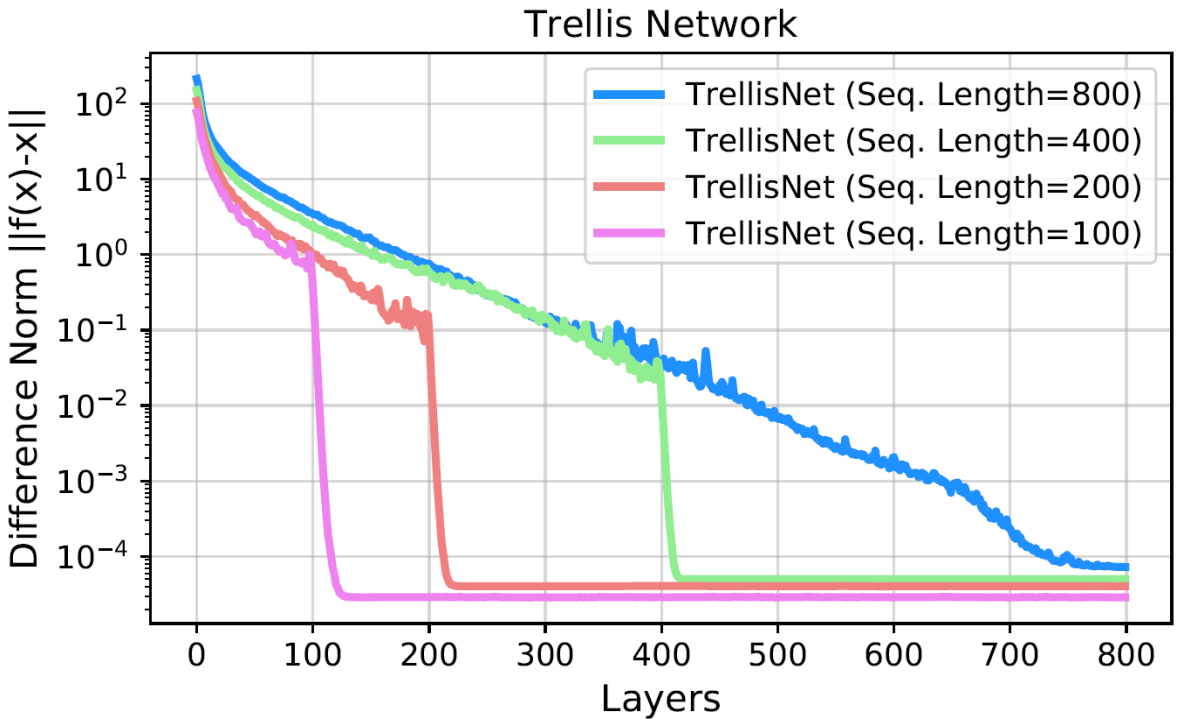
DEQ

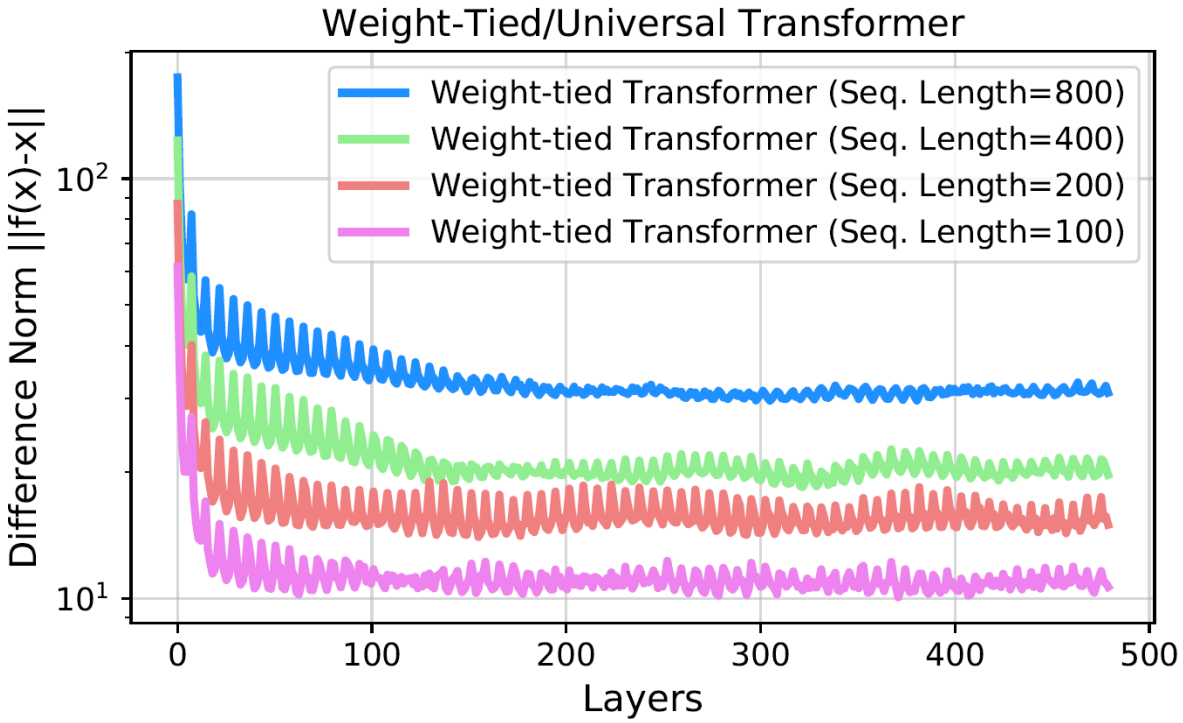
Convergence to the same point

$$\lim_{i\to\infty} \mathbf{z}_{1:T}^{[i]} = \lim_{i\to\infty} f_\theta\left(\mathbf{z}_{1:T}^{[i]}; \mathbf{x}_{1:T}\right) \equiv f_\theta\left(\mathbf{z}_{1:T}^{\star}; \mathbf{x}_{1:T}\right) = \mathbf{z}_{1:T}^{\star}$$

Equilibrium Point

Empirical Evidence





A tendency of layers to converge

Is it useful?

- Imagine that the network converges to the **equilibrium point**
- Can get equilibrium point just applying a layer many times
- How to do backprop?

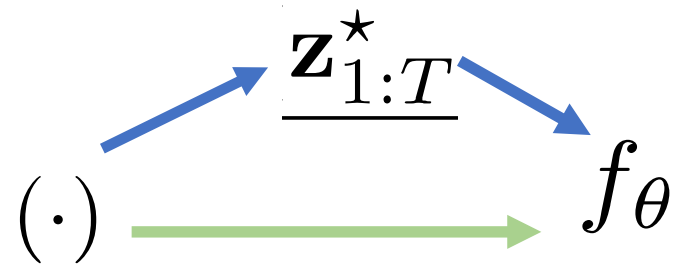
$$\frac{\partial \mathbf{z}^\star_{1:T}}{\partial(\cdot)} \; - ? \qquad f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T}) = \mathbf{z}^\star_{1:T}$$

Equilibrium equation

- Let $(\cdot)$ be a parameter of $f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})$ *(for instance, θ or $\mathbf{x}_{1:T}$)*

Implicit differentiation:

$$\frac{\partial \mathbf{z}^\star_{1:T}}{\partial(\cdot)} = \frac{\partial f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\partial(\cdot)}$$



$$\frac{\partial \mathbf{z}^\star_{1:T}}{\partial(\cdot)} = \frac{\mathrm{d} f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\mathrm{d}(\cdot)} + \frac{\partial f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\partial \mathbf{z}^\star_{1:T}} \frac{\partial \mathbf{z}^\star_{1:T}}{\partial(\cdot)}$$

$$\left(I - \frac{\partial f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\partial \mathbf{z}^\star_{1:T}} \right) \frac{\partial \mathbf{z}^\star_{1:T}}{\partial(\cdot)} = \frac{\mathrm{d} f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\mathrm{d}(\cdot)}$$

$$g_\theta(\mathbf{z}^\star_{1:T}) = f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T}) - \mathbf{z}^\star_{1:T}$$

$$J_{g_\theta}\big|_{\mathbf{z}^\star_{1:T}} = -\left(I - \frac{\partial f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\partial \mathbf{z}^\star_{1:T}} \right)$$

$$\frac{\partial \mathbf{z}^\star_{1:T}}{\partial(\cdot)} = -\left(J^{-1}_{g_\theta}\big|_{\mathbf{z}^\star_{1:T}} \right) \frac{\mathrm{d} f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\mathrm{d}(\cdot)}$$

(Do not backprop through all layers)

Backprop through Equilibrium Point

$$\frac{\partial \ell}{\partial (\cdot)} = \frac{\partial \ell}{\partial \mathbf{z}^\star_{1:T}} \frac{\partial \mathbf{z}^\star_{1:T}}{\partial (\cdot)} = -\frac{\partial \ell}{\partial \mathbf{z}^\star_{1:T}} \left(J^{-1}_{g_\theta} \Big|_{\mathbf{z}^\star_{1:T}} \right) \frac{\mathrm{d} f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\mathrm{d}(\cdot)}$$

SGD update:

$$\theta^+ = \theta - \alpha \cdot \frac{\partial \ell}{\partial \theta} = \theta + \alpha \frac{\partial \ell}{\partial \mathbf{z}^\star_{1:T}} \left(J^{-1}_{g_\theta} \Big|_{\mathbf{z}^\star_{1:T}} \right) \frac{\mathrm{d} f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\mathrm{d}\theta}$$

Reliable estimation of equilibrium point

- Use any black-box root finding method,

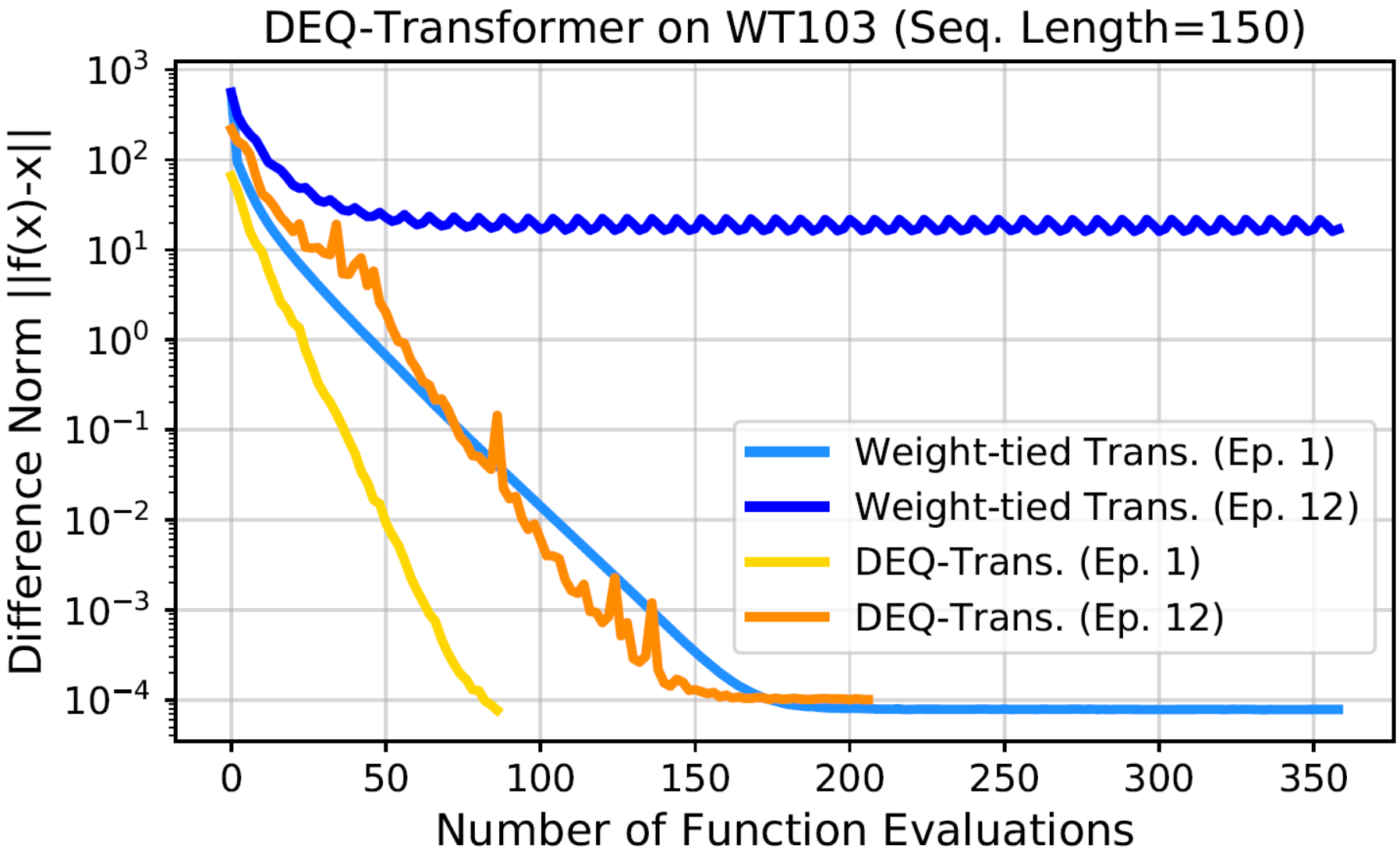
  e.g. quasi-Newton (Broyden) method (not storing full Jacobian)

$$g_\theta\left(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T}\right) = f_\theta\left(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T}\right) - \mathbf{z}^\star_{1:T} \to 0$$

- Broyden iterations:

$$\mathbf{z}^{[i+1]}_{1:T} = \mathbf{z}^{[i]}_{1:T} - \alpha B g_\theta(\mathbf{z}^{[i]}_{1:T}; \mathbf{x}_{1:T}) \quad \text{for } i = 0, 1, 2, \ldots$$

α – step size, $B \approx J^{-1}_{g_\theta}\big|_{\mathbf{z}^{[i]}_{1:T}}$ (Jacobian approximation)



DEQ-transformer finds equilibrium more reliably

Forward Pass:

$$\mathbf{z}^\star_{1:T} = \mathsf{RootFind}(g_\theta; \mathbf{x}_{1:T})$$

Backward Pass:

$$\frac{\partial \ell}{\partial (\cdot)} = -\frac{\partial \ell}{\partial \mathbf{z}^\star_{1:T}} \left(J^{-1}_{g_\theta} \Big|_{\mathbf{z}^\star_{1:T}} \right) \frac{\mathrm{d} f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\mathrm{d}(\cdot)} = -\frac{\partial \ell}{\partial h} \frac{\partial h}{\partial \mathbf{z}^\star_{1:T}} \left(J^{-1}_{g_\theta} \Big|_{\mathbf{z}^\star_{1:T}} \right) \frac{\mathrm{d} f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\mathrm{d}(\cdot)}$$

Accelerating DEQ

To compute $-\frac{\partial \ell}{\partial \mathbf{z}^\star_{1:T}}\left(J^{-1}_{g_\theta}\Big|_{\mathbf{z}^\star_{1:T}}\right)$

Solve a linear system with quasi-Newton method:

$$\left(J^{\top}_{g_\theta}\Big|_{\mathbf{z}^\star_{1:T}}\right)\mathbf{x}^\top + \left(\frac{\partial \ell}{\partial \mathbf{z}^\star_{1:T}}\right)^\top = \mathbf{0}$$

Experiments

Tasks

- **Copy Memory Task** (stress test)
- **Penn Treebank** - word-level language modeling, 888K words at training, vocabulary size of 10,000 (small language corpus)
- **WikiText-103** - vocabulary size over 260K, contains many rare words and retains punctuation, numbers, and capitalization (quite large corpus)

Results

Copy Memory Task

	Models (Size)			
	DEQ-Transformer (ours) (14K)	TCN [6] (16K)	LSTM [24] (14K)	GRU [13] (14K)
Copy Memory T=400 Loss	**3.5e-6**	**2.7e-5**	0.0501	0.0491

DEQ solves this task better than LSTM, GRU

Results

- Penn Treebank

Word-level Language Modeling w/ Penn Treebank (PTB)				
Model	# Params	Non-embedding model size	Test perplexity	Memory†
Variational LSTM [20]	66M	-	73.4	-
NAS Cell [47]	54M	-	62.4	-
NAS (w/ black-box hyperparameter tuner) [27]	24M	20M	59.7	-
AWD-LSTM [30]	24M	20M	58.8	-
DARTS architecture search (second order) [25]	23M	20M	**55.7**	-
60-layer TrellisNet (w/ auxiliary loss, w/o MoS) [7]	24M	20M	57.0	8.5GB
DEQ-TrellisNet (ours)	24M	20M	57.1	**1.2GB**

DEQ is close to SOTA, requires less memory

Results: WikiText-103

Word-level Language Modeling w/ WikiText-103 (WT103)				
Model	# Params	Non-Embedding Model Size	Test perplexity	Memory†
Generic TCN [6]	150M	34M	45.2	-
Gated Linear ConvNet [15]	230M	-	37.2	-
AWD-QRNN [29]	159M	51M	33.0	7.1GB
Relational Memory Core [34]	195M	60M	31.6	-
Transformer-XL (X-large, adaptive embed., on TPU) [14]	257M	224M	**18.7**	12.0GB
70-layer TrellisNet (+ auxiliary loss, etc.) [7]	180M	45M	29.2	24.7GB
70-layer TrellisNet with *gradient checkpointing*	180M	45M	29.2	5.2GB
DEQ-TrellisNet (ours)	180M	45M	**29.0**	**3.3GB**
Transformer-XL (medium, 16 layers)	165M	44M	24.3	8.5GB
DEQ-Transformer (medium, ours).	172M	43M	24.7	**2.7GB**
Transformer-XL (medium, 18 layers, adaptive embed.)	110M	72M	**23.7**	9.0GB
DEQ-Transformer (medium, adaptive embed., ours)	110M	70M	24.0	3.9GB
Transformer-XL (small, 4 layers)	139M	4.9M	35.8	4.8GB
Transformer-XL (small, weight-tied 16 layers)	138M	4.5M	34.9	6.8GB
DEQ-Transformer (small, ours).	138M	4.5M	**32.4**	**1.1GB**

Requires less memory than corresponding model with gradient checkpointing

Time increase

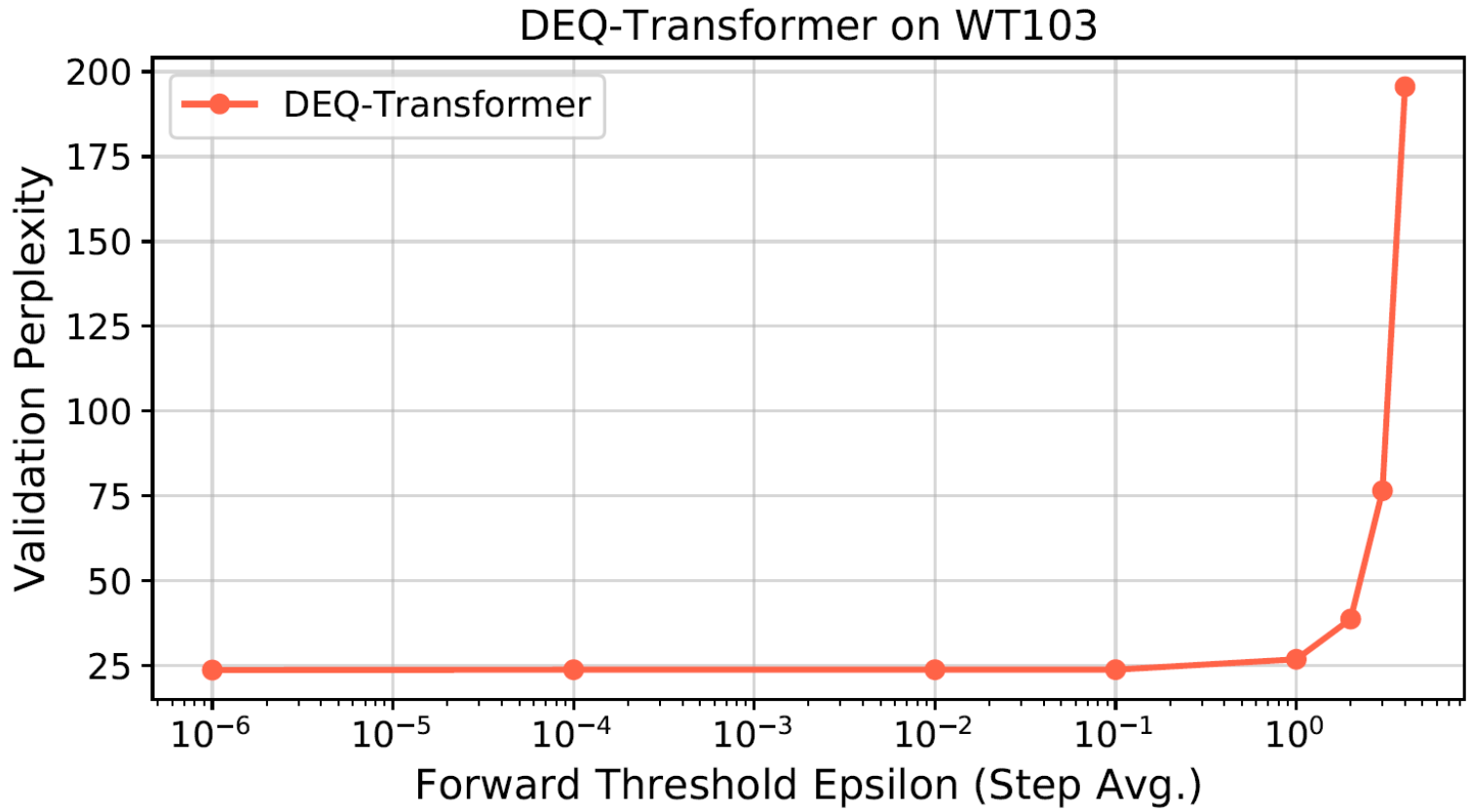
DEQ is slower than the layer-based models with competitive accuracy

DEQ / 18-layer Transformer		DEQ / 70-layer TrellisNet	
Training	Inference	Training	Inference
2.82×	1.76×	2.40×	1.64×

Results on WikiText-103

Also there is an imbalance with minibatches

Trade Broyden iterations for speed



DEQ can be accelerated by trading high tolerance for Broyden method

https://arxiv.org/pdf/1909.01377.pdf

Existence of Equilibrium

Well-Posedness Property: A square, $n \times n$ matrix A is said to be well-posed for ϕ (in short, $A \in \mathrm{WP}(\phi)$) if, for any n-vector b, the equation $x = \phi(Ax + b)$ has a unique solution.

Sufficient Condition of equilibrium point existence and uniqueness:

$\|A\|_{\alpha,\alpha} < 1$, the map $x \to \phi(Ax + b)$ is a strict contraction with respect to the l_α norm

ReLU, Sigmoid, LayerNorm, … are contractive

https://arxiv.org/pdf/1908.06315.pdf

Universality

$$\mathbf{z}^{[2]} = \sigma^{[1]}(W^{[1]}\mathbf{x} + \mathbf{b}^{[1]})$$

$$\mathbf{z}^{[i+1]} = \sigma^{[i]}(W^{[i]}\mathbf{z}^{[i]} + \mathbf{b}^{[i]}) \quad i = 2, \ldots, k$$

$$\tilde{\mathbf{z}}^{[i+1]} = \sigma(W_z\tilde{\mathbf{z}}^{[i]} + W_x\mathbf{x} + \tilde{\mathbf{b}}), \quad i = 1, \ldots, k.$$

Proof:

$$W_z = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 \\ W^{[2]} & 0 & 0 & \ldots & 0 & 0 \\ 0 & W^{[3]} & 0 & \ldots & 0 & 0 \\ 0 & 0 & W^{[4]} & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & W^{[k]} & 0 \end{bmatrix}, \qquad \tilde{\mathbf{z}} = \begin{bmatrix} \mathbf{z}^{[2]} \\ \mathbf{z}^{[3]} \\ \vdots \\ \mathbf{z}^{[k+1]} \end{bmatrix}$$

$$W_x = \begin{bmatrix} W^{[1]} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \tilde{\mathbf{b}} = \begin{bmatrix} \mathbf{b}^{[1]} \\ \mathbf{b}^{[2]} \\ \vdots \\ \mathbf{b}^{[k]} \end{bmatrix}, \quad \sigma = \begin{bmatrix} \sigma^{[1]} \\ \sigma^{[2]} \\ \vdots \\ \sigma^{[k]} \end{bmatrix}$$

Other applications of implicit layers

- ”SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver” (Po-Wei Wang, Priya L. Donti, Bryan Wilder, Zico Kolter, 2019)
- “OptNet: Differentiable Optimization as a Layer in Neural Networks”. (Brandon Amos, J. Zico Kolter, 2018)

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

$$z_{i+1} = \underset{z}{\text{argmin}} \ \frac{1}{2} z^T Q(z_i) z + q(z_i)^T z$$
$$\text{subject to} \ \ A(z_i) z = b(z_i)$$
$$G(z_i) z \leq h(z_i)$$

https://arxiv.org/pdf/1703.00443.pdf

Conclusions

- DEQ – a memory efficient model for sequential data
- Backprop directly through equilibrium point
- Every feed-forward deep model can be made implicit
- Further theoretical research is required

References

- https://arxiv.org/pdf/1909.01377.pdf - Deep Equilibrium Models
- https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf - Transformer
- https://arxiv.org/pdf/1807.03819.pdf - Universal Transformer
- https://arxiv.org/pdf/1810.06682.pdf - Trellis Networks
- https://github.com/cybertronai/gradient-checkpointing - Gradient Checkpointing
- https://arxiv.org/abs/1806.07366 - Neural ODEs
- https://arxiv.org/pdf/1908.06315.pdf - Implicit Deep Learning
- https://arxiv.org/pdf/1703.00443.pdf - OptNet
- https://arxiv.org/pdf/1905.12149.pdf - SATNet