"DEQ"

Deep Equilibrium Models

Shaojie Bai

Carnegie Mellon University

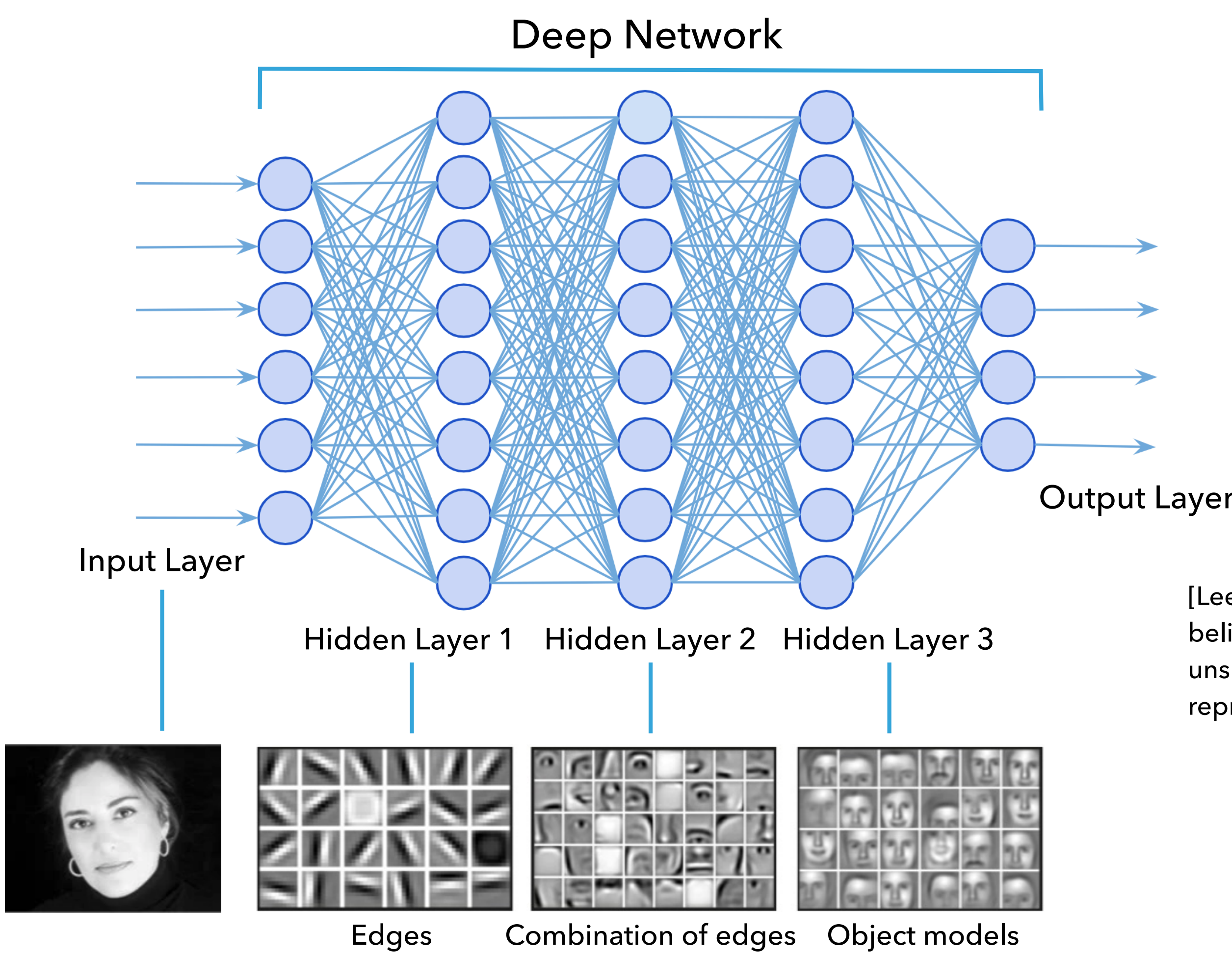
joint work with **J. Zico Kolter** (CMU/Bosch) and **Vladlen Koltun** (Intel)

NeurIPS 2019

TL;DR: One (implicit) layer is all you need, with proof.

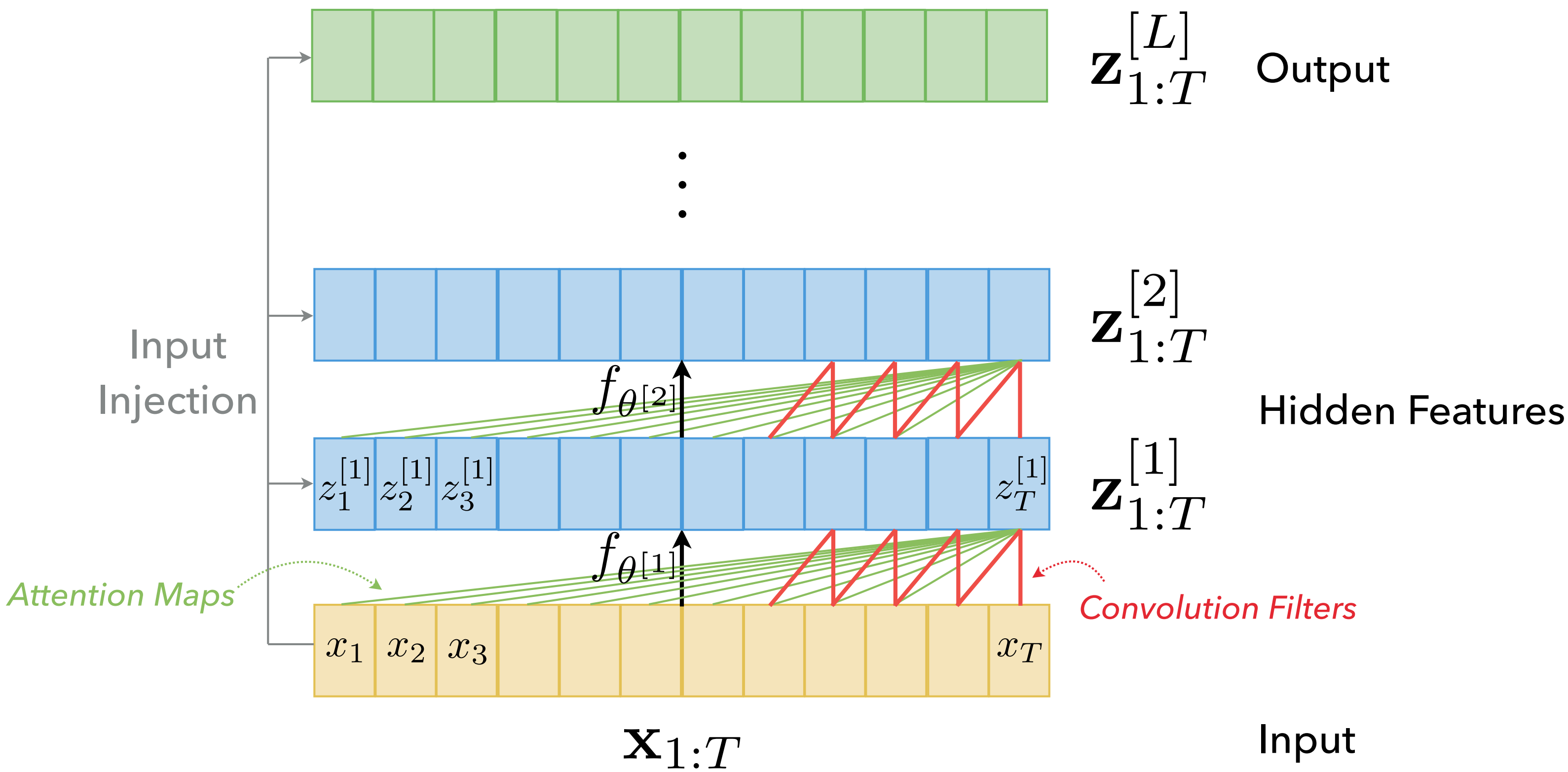
"Deep Learning Philosophy"

The story we all tell: deep learning algorithms build hierarchical models of input data, where earlier layers extract "simple" features and later layers create high-level abstractions of the data. A famous example:



[Lee, et al., "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations", ICML 2009]

Deep (Feed-Forward) Nets on Sequences



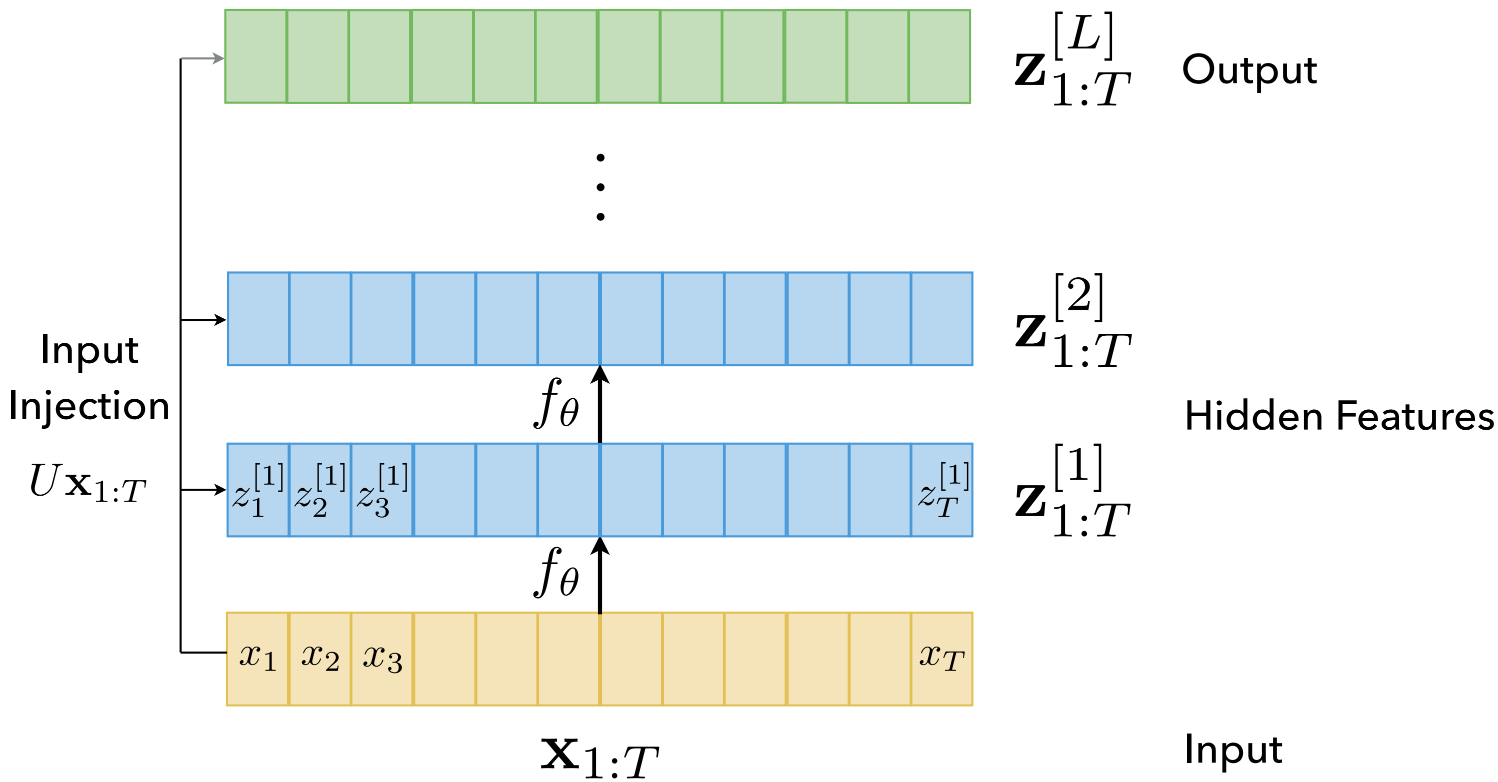
An L-layer conventional deep network on a sequence (with length T):

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

E.g.: (Temporal) Convolution, (Self-Attention) Transformer

Weight-Tied, Input Injected Networks

(Some works call this "recurrently stacked")



An L-layer weight-tied, input-injected deep network on a sequence (with length T):

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

Isn't Weight-Tying a Big Restriction?

Theoretically, no…

- **Theorem** [informal]: Any deep feedforward network can be represented by a weight-tied, input-injected network of equivalent depth.

Input

$$\begin{aligned}\mathbf{z}^{[1]} &= \sigma(W_1\mathbf{x} + b_1)\\ \mathbf{z}^{[2]} &= \sigma(W_2\mathbf{z}^{[1]} + b_2)\\ \mathbf{z}^{[3]} &= \sigma(W_3\mathbf{z}^{[2]} + b_3)\end{aligned} \iff \begin{bmatrix}\mathbf{z}^{[1]}\\ \mathbf{z}^{[2]}\\ \mathbf{z}^{[3]}\end{bmatrix} = \sigma\left(\begin{bmatrix}0 & 0 & 0\\ W_2 & 0 & 0\\ 0 & W_3 & 0\end{bmatrix}\begin{bmatrix}\mathbf{z}^{[1]}\\ \mathbf{z}^{[2]}\\ \mathbf{z}^{[3]}\end{bmatrix} + \begin{bmatrix}W_1\\ 0\\ 0\end{bmatrix}\mathbf{x} + \begin{bmatrix}b_1\\ b_2\\ b_3\end{bmatrix}\right)$$

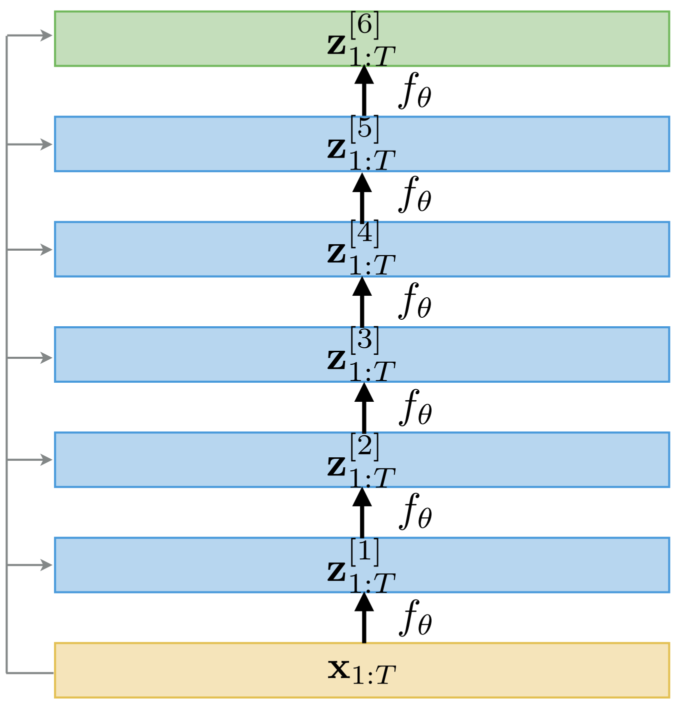
(Apply this **three** times to $\begin{bmatrix}\mathbf{z}^{[1]} & \mathbf{z}^{[2]} & \mathbf{z}^{[3]}\end{bmatrix}^\top = \mathbf{0}$)

Empirically, no…

- Weight-tied models with parameter count smaller (but on the same scale) achieve performances just as good; see, e.g., TrellisNet [Bai et al., ICLR 2019], Universal Transformer [Dehghani et al., ICLR 2019], (and more recently) ALBERT [Anonymous, submitted to ICLR 2020].

Equilibrium Points and the DEQ Model

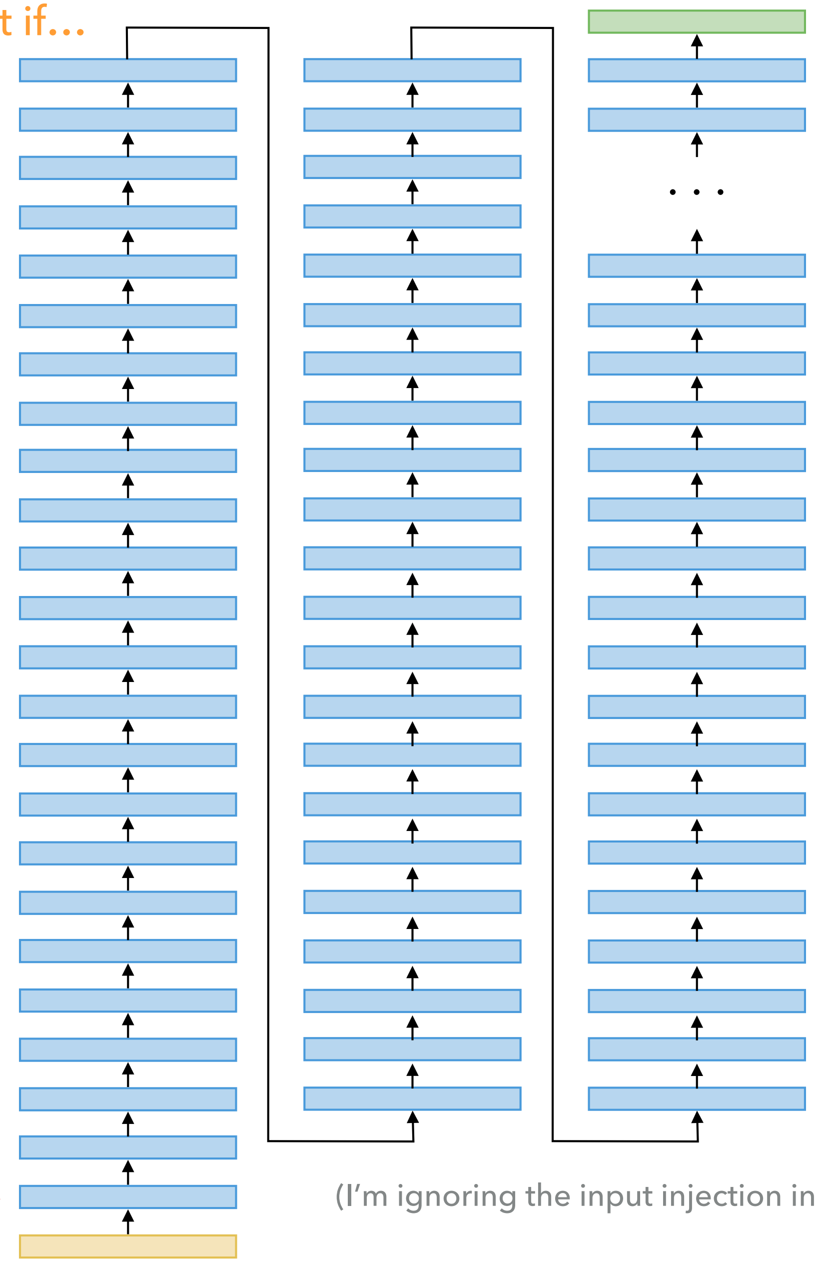
of layers typically are hand-picked by model designers:



Input

(e.g., a 6-layer weight-tied network)

What if…



Not possible in practice, due to memory/computation limits

(I'm ignoring the input injection in this figure)

Equilibrium Points and the DEQ Model

We now can think of a "deep" network as repeated applications of a function:

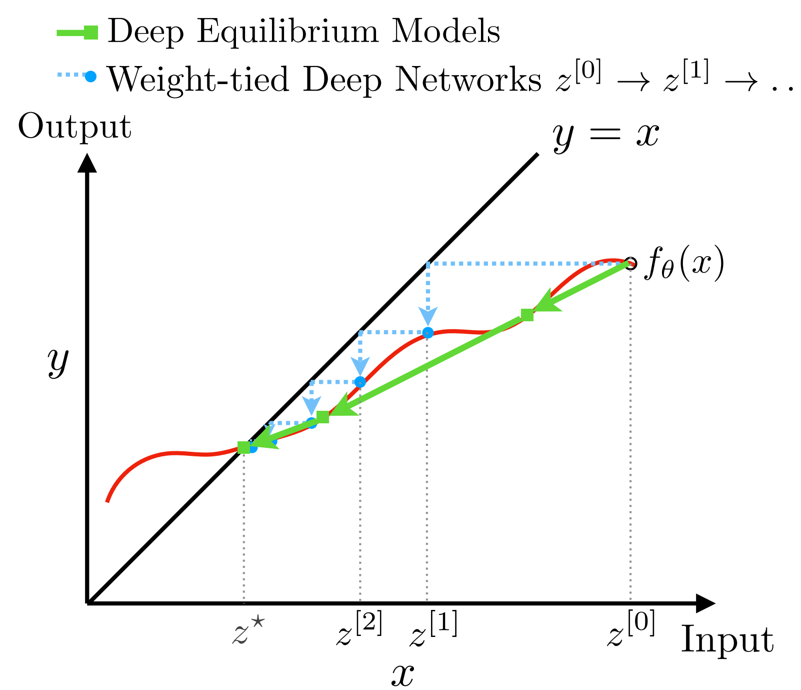
$$\mathbf{z}^{[i+1]} = f(\mathbf{z}^{[i]}; \mathbf{x})$$

In practice, after these types of models converge to an equilibrium point (i.e., an "infinite depth" network):

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

(Stop by our poster for more empirical evidence :-))

Deep Equilibrium (DEQ) Model: directly find this equilibrium/stable point via root-finding (e.g., Broyden's method), rather than just iterating the forward model, and apply implicit differentiation for backpropagation.



Overview of DEQ Approach

Define a single layer $f_\theta(\mathbf{z}_{1:T}; \mathbf{x}_{1:T})$.

Forward pass: Given an input $\mathbf{x}_{1:T}$, compute the equilibrium point $\mathbf{z}^\star_{1:T}$, such that

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

(via any black-box root solver; such as Quasi-Newton methods)

Backward pass: Implicitly differentiate through the equilibrium state to form gradients:

$$\frac{\partial \ell}{\partial (\cdot)} = -\frac{\partial \ell}{\partial \mathbf{z}^\star_{1:T}} \left(\frac{\partial f_\theta}{\partial \mathbf{z}^\star_{1:T}} - I \right)^{-1} \frac{\partial f_\theta}{\partial (\cdot)}$$

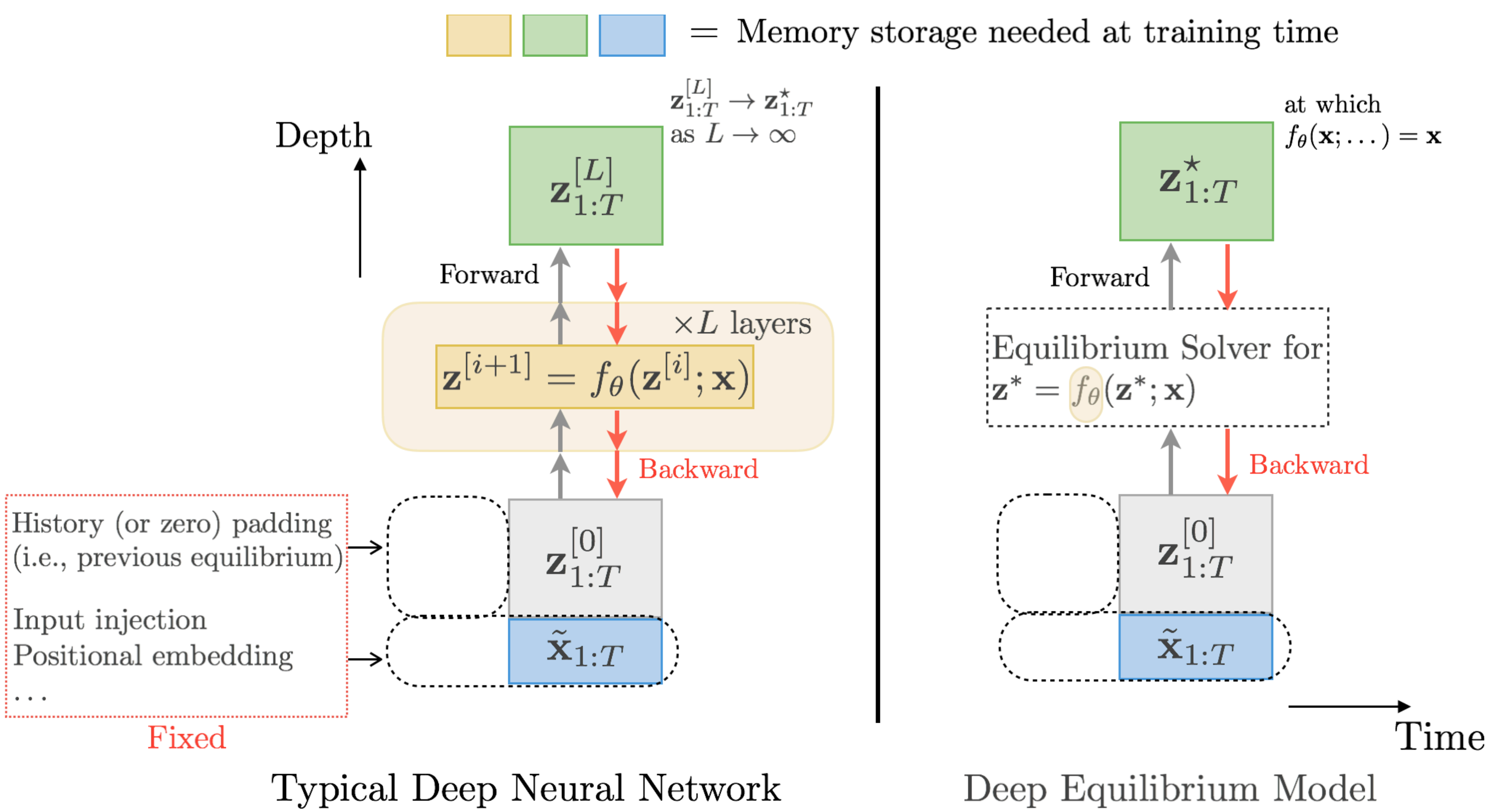
Jacobian **at the equilibrium** (i.e., output)

Gradient of one layer

More details/proof in the paper.

Overview of DEQ Approach

To compare conventional deep networks with DEQ:



Overview of DEQ Approach

Memory Footprint of DEQs?

Forward pass: black-box root solving (e.g., fast Quasi-Newton methods)

Backward pass: One-step multiplication with the Jacobian at equilibrium

This implies constant memory consumption: only need to store $\mathbf{x}_{1:T}$, $\mathbf{z}^{\star}_{1:T}$, and θ (no growth at all with depth), for training an "infinite-depth" network!

Stacking DEQs of different functions?

- **Theorem** [informal] (*Universality* of "single-layer" DEQs):

Given transformations $f_{\theta^{[1]}}$ and $\nu_{\theta^{[2]}}$ (from potentially different function classes), there exists a transformation Γ_{Θ} such that $\mathrm{DEQ}(\Gamma_{\Theta}; \mathbf{x}_{1:T}) = \mathrm{DEQ}(\nu_{\theta^{[2]}}; \mathrm{DEQ}(f_{\theta^{[1]}}; \mathbf{x}_{1:T}))$.

Runtime of DEQs?

Typically ~2-2.5x slower to train, ~1.7x slower at inference (as root-finding takes longer than iterating a fixed number of forward layers)

DEQs for Sequence Modeling

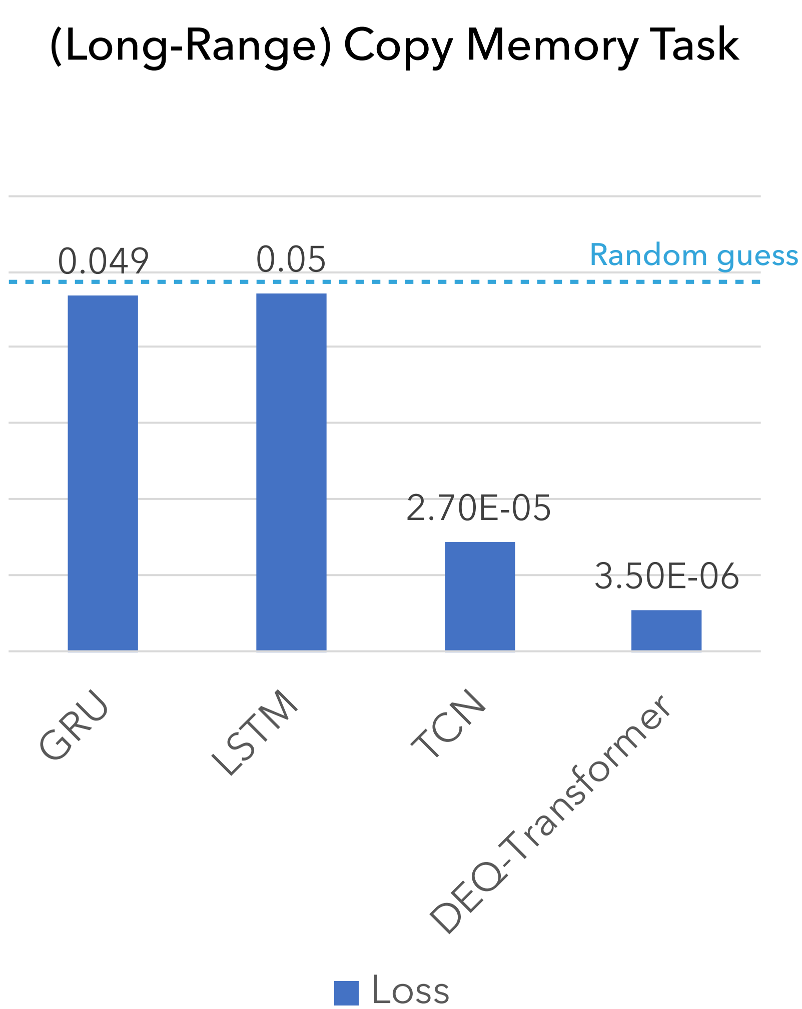
- One can easily extend the methods above to create “infinite-layer” versions of all common sequence modeling architectures.

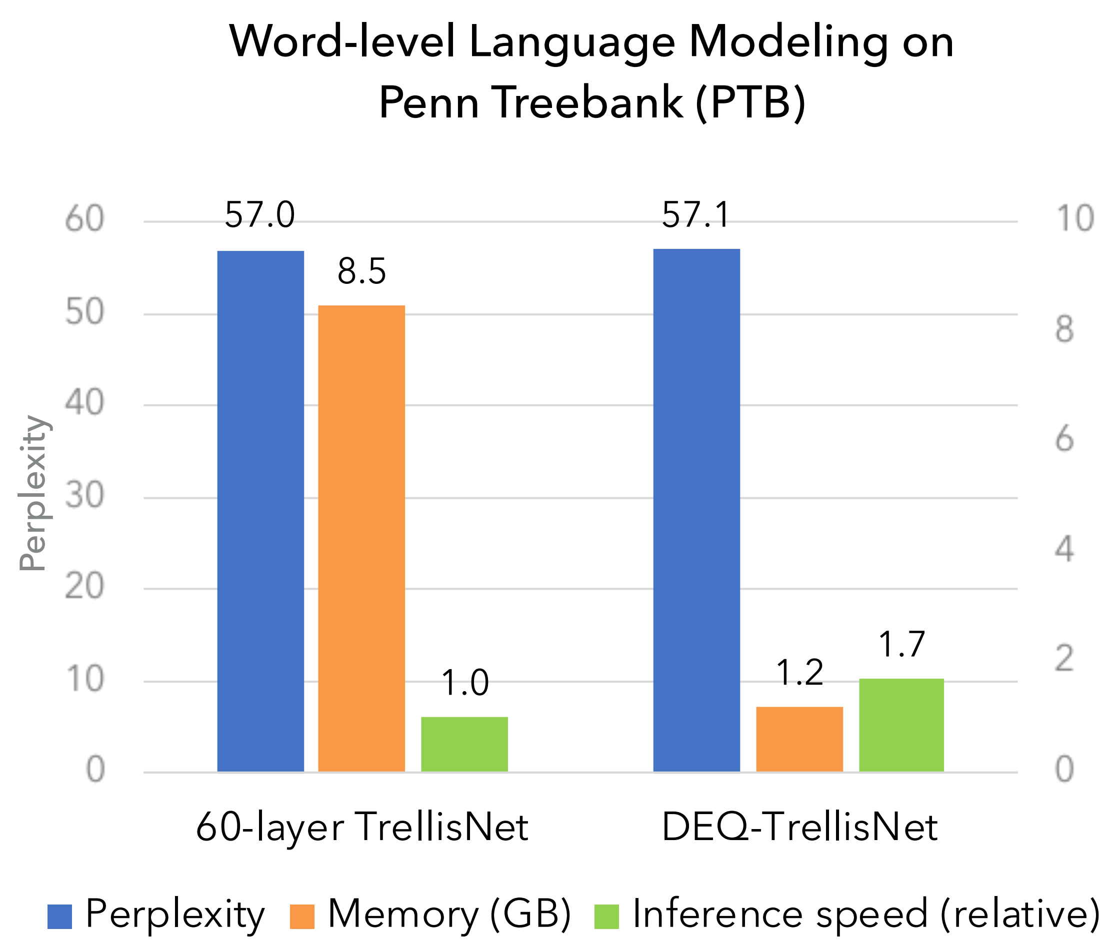
- We specifically provide two *instantiations of DEQ* based on two very different SOTA sequence modeling architectures:

  1) **DEQ-TrellisNet**: equilibrium version of TrellisNet architecture [Bai et al., ICLR 2019], a type of weight-tied temporal convolutions that generalizes RNNs

  2) **DEQ-Transformer**: equilibrium version of Transformer architecture [Vaswani et al., NeurIPS 2017], with weight-tied multi-head self-attention [Dehghani et al., ICLR 2019]

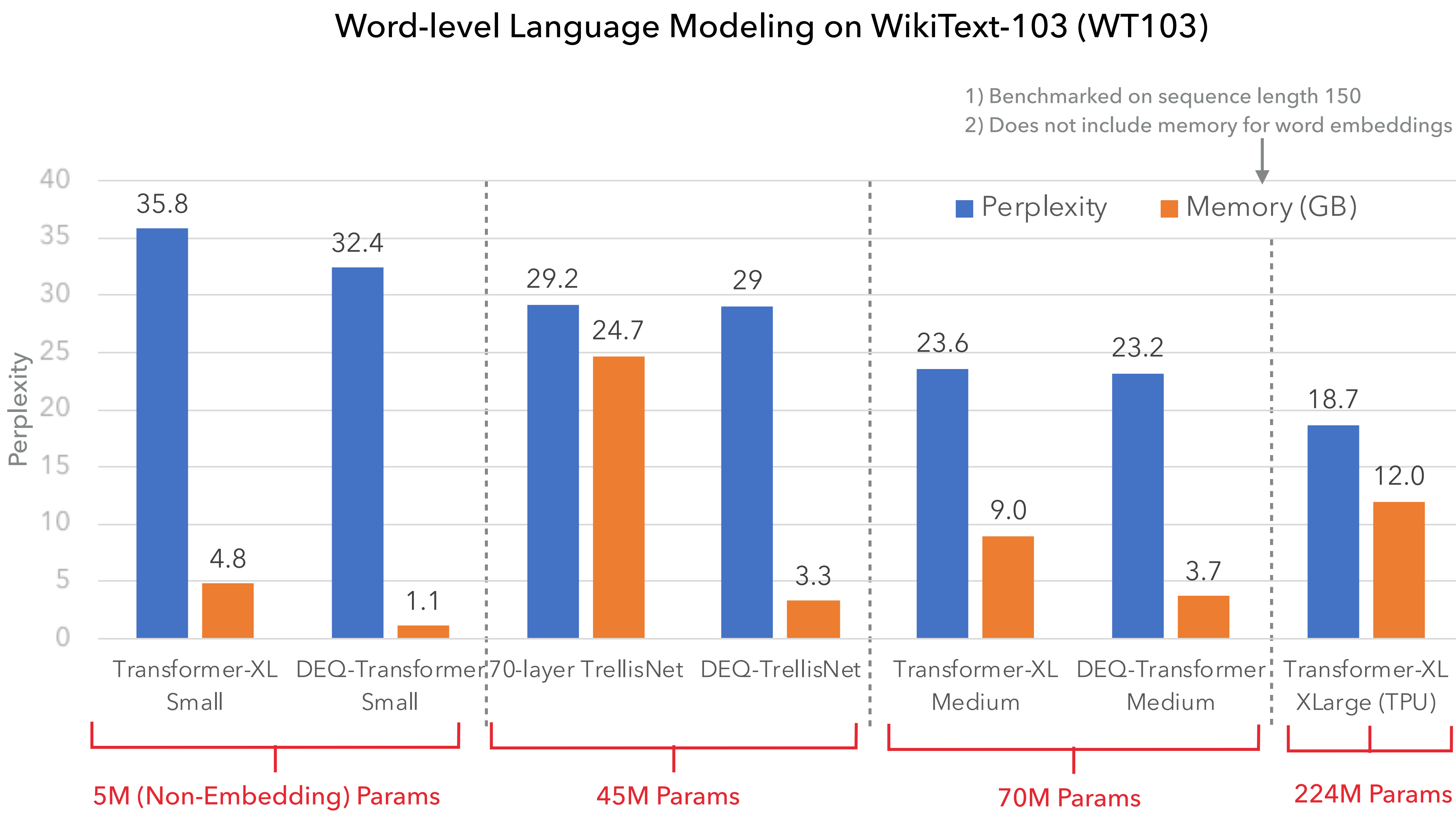
Small-Scale Benchmarks





1) Benchmarked on sequence length 150
2) Does not include memory for word embeddings

Large-Scale Benchmarks



Summary, Thoughts and Challenges

- DEQ represents the largest-scale practical application of implicit layers in deep learning of which we are aware.

- DEQ computes an “infinite-depth" network. DEQ’s **forward pass** relies on a direct root solving; its **backward pass** relies only on the equilibrium point, not on any of the intermediate “hidden features". Memory needed to train DEQ is therefore constant (i.e., equivalent to that of 1 layer).

- DEQ performs competitively with SOTA architectures, but with up to 90% reduction in memory cost.

- How should we understand depth in deep networks?

- Let the objective of a model be implicitly defined (e.g., “the equilibrium")?

Interested in DEQ? Stop by our poster at #XXX (right after this talk) ;-)

Shaojie Bai | shaojieb@cs.cmu.edu | https://github.com/locuslab/deq | @shaojieb