"DEQ"

Deep Equilibrium Models

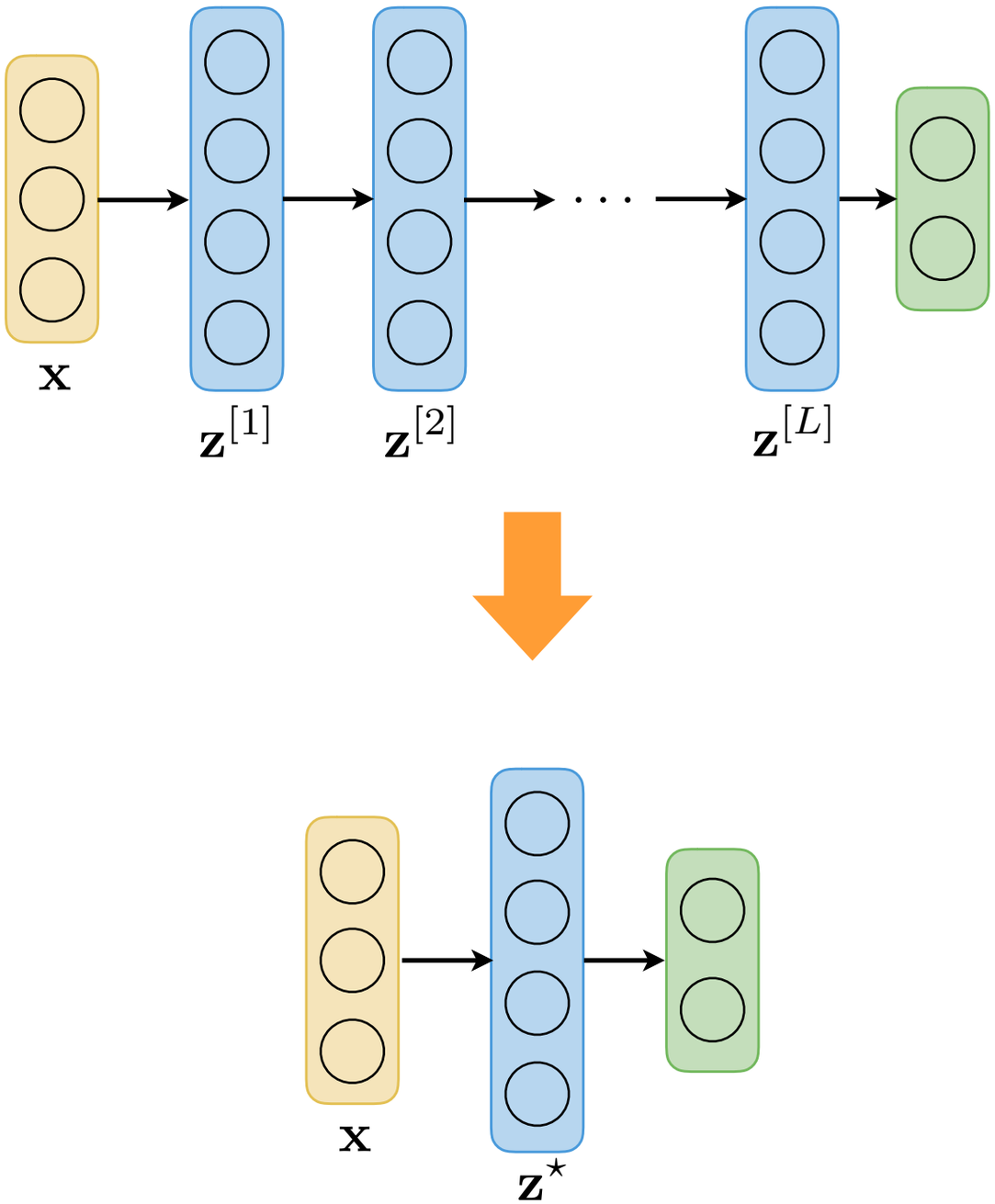
Shaojie Bai
Carnegie Mellon University

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

NeurIPS 2019

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

Outline of This Talk

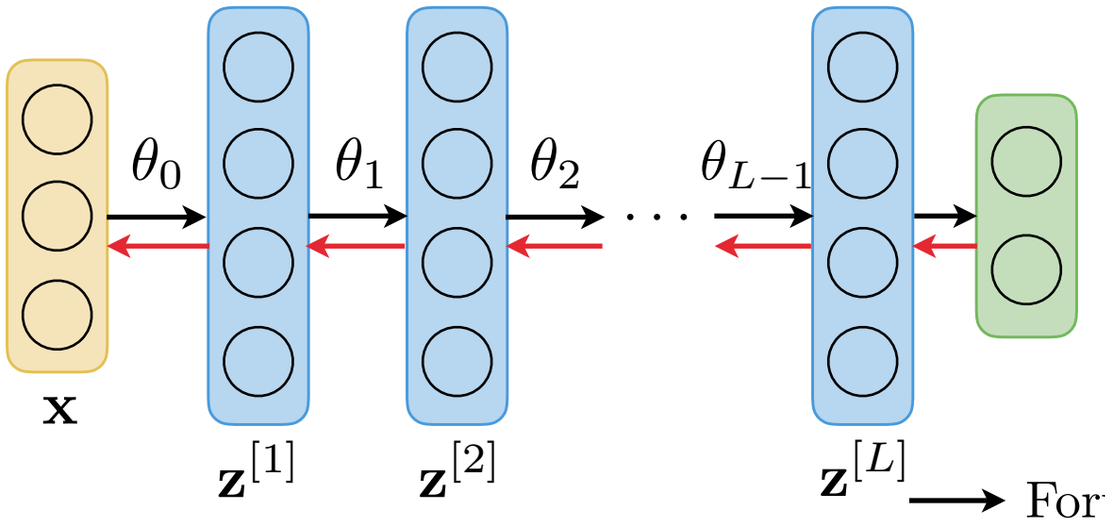


We can replace *many classes of* deep models with a single layer, keep the number of parameters the same, and lose no **representational capacity**.

Requires us to (re-)consider deep networks implicitly, with an approach that we call the **deep equilibrium (DEQ) model**.

Works as well (or better) than existing models on large-scale sequence tasks while using only constant memory.

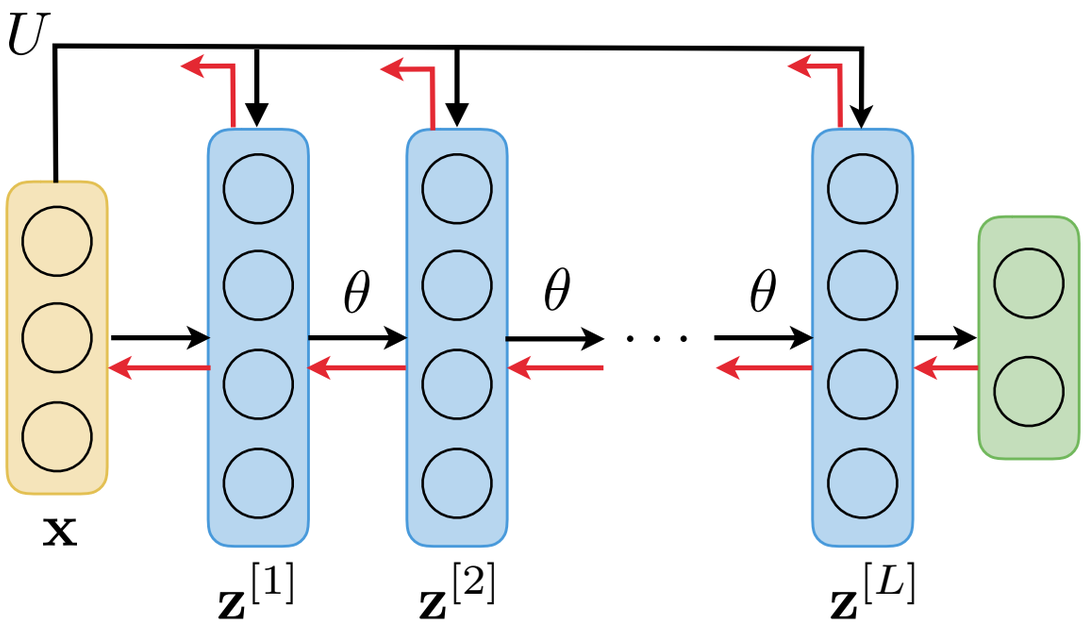
Weight-Tied, Input-Injected Networks



Traditional layer:

$$\mathbf{z}^{[i+1]} = f_{\theta_i}(\mathbf{z}^{[i]}) = \sigma(W_i \mathbf{z}^{[i]} + b_i)$$

(just a simple example)



Weight-tied input-injected layer:

$$\mathbf{z}^{[i+1]} = f_\theta(\mathbf{z}^{[i]}; \mathbf{x}) = \sigma(W\mathbf{z}^{[i]} + U\mathbf{x} + b)$$

Isn't weight-tying a big restriction?

- **Theoretically, no**: We show that any deep feedforward network can be represented by a weight-tied, input-injected network of equivalent depth.

- **Empirically, no**: The (many) recent successes of weight-tied models: TrellisNet [Bai et al., ICLR 2019], Universal Transformer [Dehghani et al., ICLR 2019], ALBERT [Lan et al., preprint].

Equilibrium Points, and the DEQ Model

We now can think of a deep network as *repeated* applications of some function

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

In practice (a bit more on this point shortly), after these types of models converge to an equilibrium point (i.e., an “infinite depth" network)

$$\mathbf{z}^\star = f_\theta(\mathbf{z}^\star; \mathbf{x})$$

Deep Equilibrium (DEQ) Models: Find this equilibrium point directly via root-finding (e.g., Newton/quasi-Newton methods) rather than iterating the forward model. Backpropagate via implicit differentiation.

A Formal Summary of the DEQ Approach

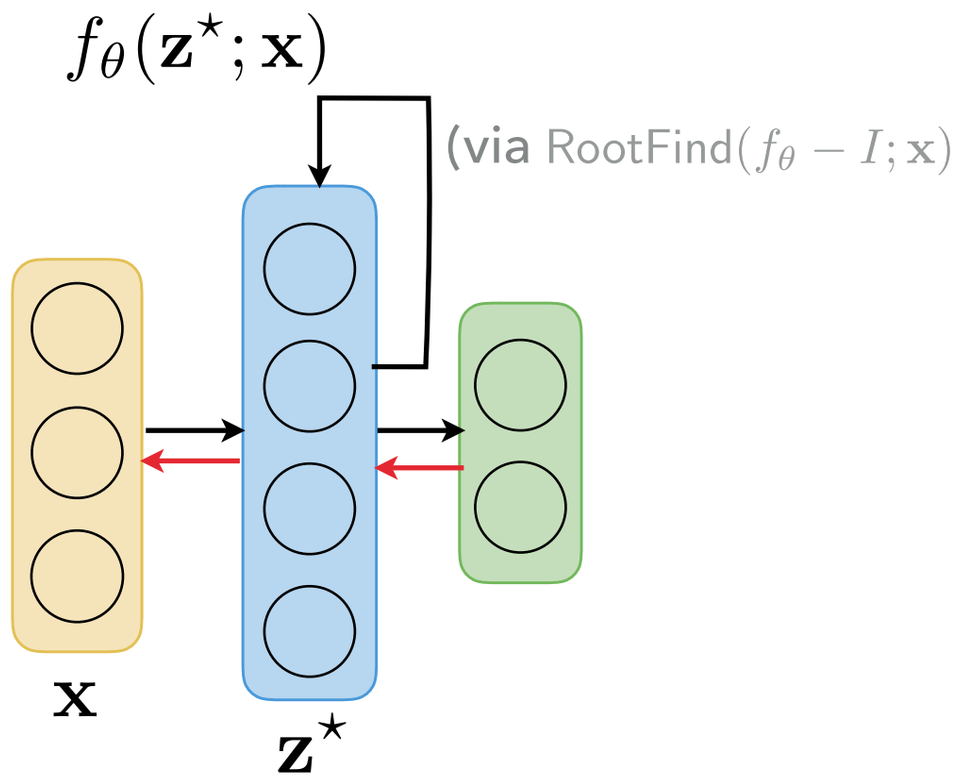
Define a single layer $f_\theta(\mathbf{z}; \mathbf{x})$.

Virtually always exists in practice (examples later)

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

$$f_\theta(\mathbf{z}^\star; \mathbf{x}) - \mathbf{z}^\star = 0$$

(via any black-box root solver; e.g. Broyden's method)



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

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

Gradient of one layer

Jacobian **at the equilibrium**

FAQs

Q: Is DEQ related to the decade-old *attractor network*, and the *recurrent backprop* (RBP) ideas?

- **Yes**! Our main contributions here are conceptual and empirical: 1) We advocate for replacing general, modern, highly structured networks with single-layer equilibrium models, not using simple recurrent cells; and 2) We demonstrate that with these networks, the method can achieve SOTA performance with vast reduction in memory.

Q: Why not stack *these* deep equilibrium "implicit" layers (with potentially different functions)?

- **No**! Stacked DEQs can be equivalently represented as a single (wider) DEQ; i.e., “deep” DEQs doesn’t give you more; it’s only a matter of designing f_θ.

  Intuitively, $\exists\ \Gamma_\Theta \text{ s.t. } \text{DEQ}_{\Gamma_\Theta} = \text{DEQ}_{h_{\theta_2}} \circ \text{DEQ}_{f_{\theta_1}}$

FAQs

Q: What are the relative time/memory tradeoffs?

- **Typically ~2-2.5x slower to train, ~1.5-2x slower for inference** (root finding takes slightly longer than iterating a small fixed # of forward steps).

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

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

- **Constant memory consumption**: no need to store any intermediate value (i.e., no growth at all with "depth"; O(1)). Only need to store $\mathbf{x}, \mathbf{z}^\star, \theta$.

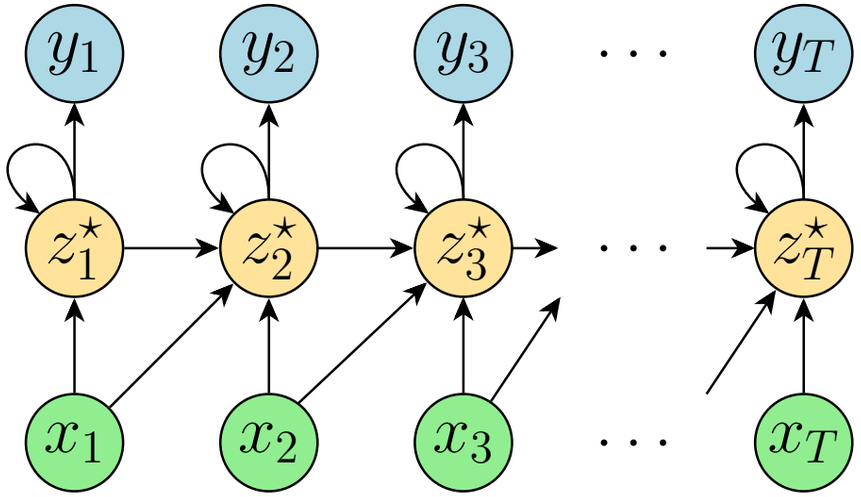
DEQs for Sequence Modeling

- One can easily extend the methods above to create DEQ 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

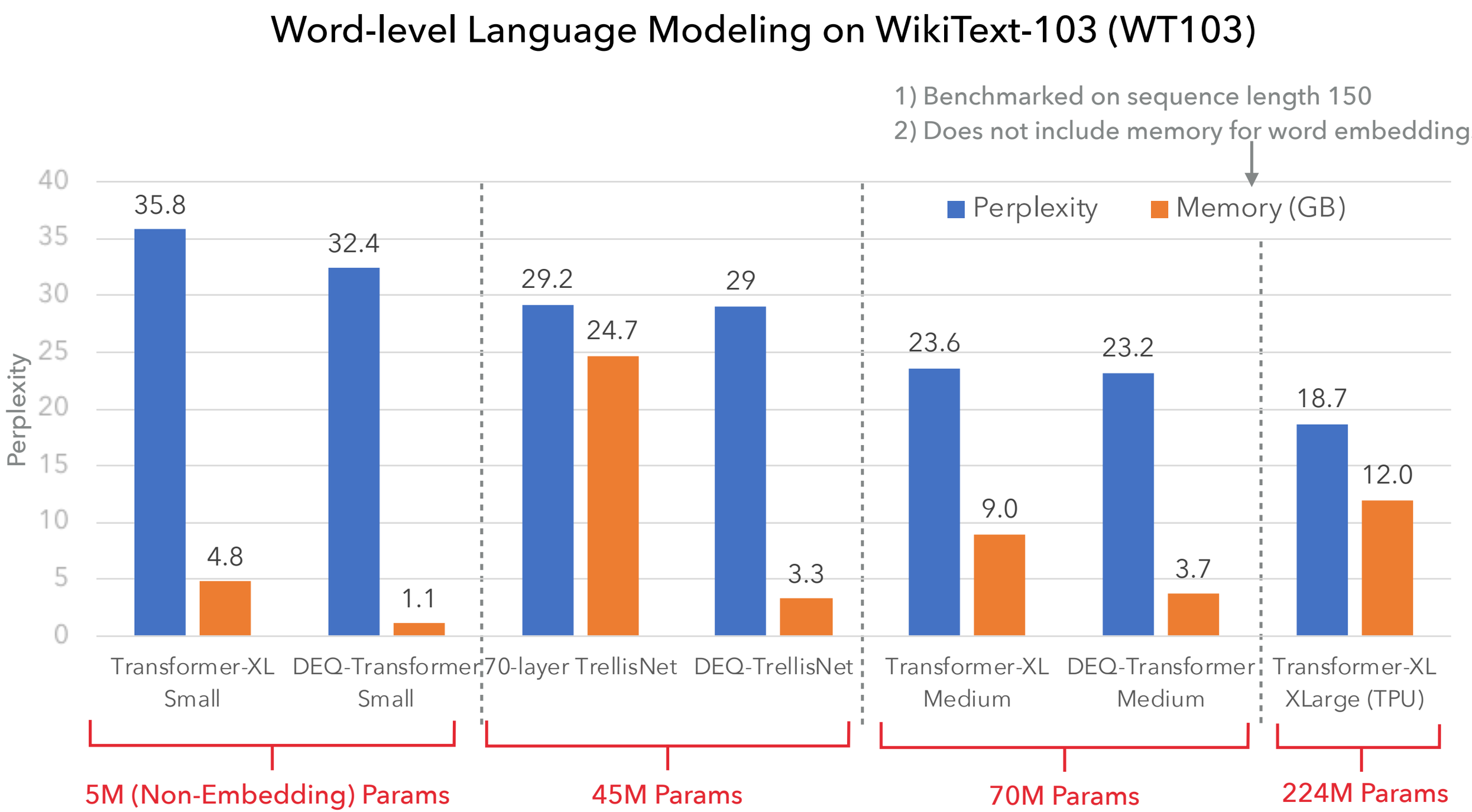
2) **DEQ-Transformer**: equilibrium version of Transformer architecture [Vaswani et al., NIPS 2017], with weight-tied multi-head self-attention [Dehghani et al., ICLR 2019]



$$\mathbf{z}_{1:T}^\star = f_\theta(\mathbf{z}_{1:T}^\star; \mathbf{x}_{1:T}) = \text{RootFind}(g_\theta; \mathbf{x}_{1:T})$$

More details in the paper.

Large-Scale Benchmarks



More results in the paper.

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
Exhibition Hall B+C #137 (right after this talk) ;-)

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